I.J. Information Technology and Computer Science, **2015, 01, 54-66**
Published Online December 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2015.01.07

An Empirical Study on Testing of SOA based Services

Abhishek Kumar
Trinity institute of technology & research, Bhopal (M.P), India
Email: abhikumar695@gmail.com

Manindra Singh
Samsung Research Institute, Delhi, India
manindrakvfw@gmail.com

Abstract— Service-Oriented Architecture (SOA) removed the gap between software and business. Today, there is a business transformation among enterprises and they adopt a service based information technology (IT) model. So, testing is necessary for SOA based applications. This paper investigates different type of approaches and techniques that address the testing problems of SOA based services. Here we also investigate the differences between SOA and web services and traditional testing and SOA testing. Various testing levels are also discussed in detail. This paper also expresses various testing perspectives, challenges of SOA testing and review the many testing approaches and identify the problems that improve the testability of SOA based services.

Index Terms— SOA, SOA Testing, Testing Perspectives, Testing Challenges, Web Services

I. INTRODUCTION

The service-oriented architecture supports service-orientation. Service-Orientation represents a way of thinking about business and IT. SOA removed the gap between software and business. Today, there is a business transformation among enterprises and they adopt a service based IT model. SOA provides solutions for integrating diverse systems that support interoperability, loose coupling and reuse. To full-fill clients need one service invoke another service/services. It is possible that there is some evolution among these external services. This dynamic and adaptive nature of SOA makes major concerns about its reliability and fault-free implementation. So, testing is necessary for SOA based application. However, the SOA testing approach is completely different from traditional testing approach. In traditional testing approach test location is centralized and testing is mostly done by software provider. There is offline regression testing and static test case profiling. But, SOA testing approach supports collaborative testing. Here, verification is done among the service providers, service brokers and clients in a collaborative way. In SOA testing, testing location should be remote, distributed, multi-phase and multi-agent. SOA testing supports online regression testing where data collected dynamically. Reliability is ensured by dynamic profiling and group testing. Service-oriented architecture is composed of loosely coupled, discoverable, reusable and interoperable services.

SOA is an approach for designing managing and deploying systems that represent reusable business functionality. Web services are used to implement the SOA in which service interface describes using the *web service description language* (WSDL) and *extensible markup language* (XML). The message is transmitted using *simple object access protocols* (SOAP) over *hypertext transfer protocol* (HTTP) [33]. The web service is based on open standards such as HTTP and XML-based protocols including SOAP and WSDL, Web services are hardware, programming language, and operating system independent [34]. Web services are formally and fully described by an XML-based WSDL document. The best way to approach a WSDL document is to understand that different XML elements take responsibility for describing different levels of detail. For example, the <message> element is a detailed listing of the types that factor into a given message. On the other hand, the <operation> element simply lists the messages that factor into a given operation without going into any detail as to what these messages look like [7].

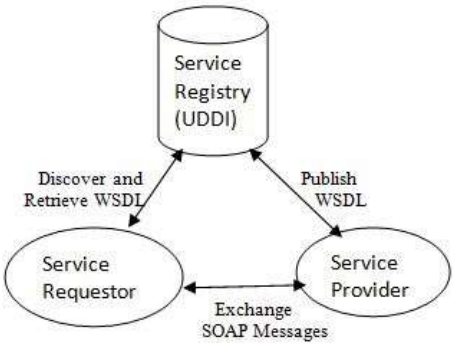

Fig. 1. Primitive SOA

WSDL documents fully describe a Web service, including the operations that it supports, the messages that it exchanges, and the data types that these messages use. An XML based protocol SOAP is used to exchange data over HTTP. SOAP is used by web services for sending messages between service provider and service consumer. Since all browsers and web servers support

HTTP so applications exchange their SOAP message regardless of their language and platform. This quality ensures the interoperability in web services. The service requester sends a query to *universal description, discovery and integration* (UDDI) registry for searching services and a WSDL file is obtained for the service the requester wants to utilize [34]. UDDI is a standard sponsored by *Organization for the Advancement of Structured Information Standards* (OASIS). UDDI is a specification for creating an XML-based registry that lists information about businesses and the Web services they offer. UDDI provides businesses a uniform way of listing their services and discovering services offered by other organizations [5]. When we use an e-commerce website to place our order, we can easily understand the concept of services. During purchasing through online shopping we have to supply our credit card information. This credit card information is validated by the outside service vendor. Once the order has been finalized, the e-commerce company deliver our order and for this they have to co-ordinate with the shipping service vendor.

Service-Oriented Architecture (SOA) is an architectural style that uses services as a building block and to facilitate enterprise integration and reuse components through loose coupling. In SOA service is defined by an interface and it makes service as a platform independent. Services can be of three types: entity service, business service and process service. Interfaces should be cohesive in nature so that they relate to each-other in the module context. Modular compos ability, modular decomposability, modular continuity, modular understandability and modular protection are the principle of modularity. The interface contains operation name, input-output parameter. Operation defined in interface carries out business functions by operating on business objects. The decomposition of systems into services also have been provided by already established concepts like *Common Object Request Broker Architecture* (CORBA) and *Distributed Component Object Model* (DCOM). SOA adaptation allows the developer to overcome many distributed challenges such as:- transaction management, security policies, application integration etc.

SOA support service-orientation principles and hence interaction between services much easier. We use SOA paradigm for organizing and utilizing distributed capabilities of different domains and the technology that they use. SOA supports the service discovery concept. Service consumers send a request to a registry for a service that they needs. SOA support interoperability that is achieved by protocol and data format of potential clients and service's current clients. A system that uses SOA are self-healing meaning that such type of systems having the ability to recover from errors without human intervention during execution. The system that builds upon SOA is completely different from the systems that builds upon object-oriented model. We can understand the difference between this two in the following way:-

Table 1. Difference Between Object-Oriented Development and SOA Development.

S. No.	Object-Oriented Technology	SOA Development
1.	There is a tight coupling in object-oriented model. If we want to change in a single point we need to implement in a large portion.	In SOA based development services are independent of each-other. Services can also be reused many times with different set of processes.
2.	In object - oriented development there is a complicated structure with various dependencies and hierarchical models.	System that use SOA policies having simple architectural style and the modeling is also performed at run time.
3.	In object-oriented development static model is developed at the design time.	In SOA based development there is an independent dynamic application architecture.

A. Characteristics of Contemporary SOA

Following are the characteristics of contemporary SOA [41]:

a) SOA increase quality of service.
b) SOA is fundamentally autonomous.
c) SOA supports vendor diversity.
d) SOA fosters intrinsic interoperability.
e) SOA promotes discovery.
f) SOA promotes federation.
g) SOA supports a service-oriented business modeling paradigm.
h) SOA implements layers of abstraction.
i) SOA promotes loose coupling throughout the enterprise.
j) SOA promotes organizational agility.
k) SOA emphasizes extensibility.
l) SOA is an evolution.

B. Difference between Primitive SOA and Contemporary SOA

Table 2. Difference between Primitive SOA and Contemporary SOA

S. No.	Primitive SOA	Contemporary SOA
1.	Primitive SOA is a baseline SOA	Represent the extension of the primitive SOA
2.	Primitive SOA provides guideline realize by all vendors.	Contemporary SOA focus on securing content of messages, enhancing XML/SOAP processing etc.

C. SOA and Web Services

Implementation of SOA is done by the web services. But, there is a huge difference in the concept of web services and SOA. Following are the differences between SOA and web services [38] [39] [40]:-

Table 3. Difference between SOA and Web service

S. No.	SOA	Web Services
1.	SOA is the set of architectural concept used for developing and integrating the services.	Web services can be used as reusable components that support interoperable machine-to-machine interaction over the network.
2.	SOA separates the service interface (the what) from its implementation (the how).	Web service promises the tools for the automation of business-to-business relationship over the internet.
3.	SOA is an architectural style that has been around for years.	Web services are the preferred way to realize SOA.
4.	SOA is a methodology not a standard.	The web service is a standard.
5.	SOA is a governance plan.	The web service is a conversation i.e., Communication between the services.

The rest of the paper is structured as follow. Section II explain various SOA testing concept in details. Section III gives overview of some SOA testing existing approaches. Year wise various SOA testing approaches are summarized in section IV. Section V gives conclusion of this paper.

II. SOA TESTING

This section explain various SOA Testing concept like- SOA testing perspectives, SOA testing levels and SOA testing challenges. These concepts are described as follow-

A. SOA Testing Perspectives

SOA imposes different need and challenges to different stakeholder involves in the testing activities. Stakeholder interested to make sure the originality of service behavior during its lifetime. Different testing perspectives are detailed below:

a) **Provider/Developer perspective:** Services are built by service developer and he deliver both the interface and implementation of services. In order to release a highly reliable service, service developer need to test the services. Developer derived the test cases but it may not reflect real usage scenarios. Service provider tests the services that meet the *service level agreement* (SLA) agreed upon with the customer. Black box testing technique is used by the service provider [11,14,17].

b) **Service Integrator's Perspective:** Here service testing is done during the design phase in order to explore the functional and non-functional assumptions. Existing services are used by the service Integrator to create an end-user application. The focus of Integrator perspective is on business need rather than implementation details which demand several services invocation [11,14,17].

c) **Third-Party/ Certifier Perspective:** Service Integrator employed third parties for the assessment of fault-proneness of a service. The trusted third-party helps to reduce the number of stakeholders and resources involved in the testing activities. The third-party/ certifier testing perspective is different from the Service Integrator's perspective [11,14].

d) **End-user Perspective:** Service-oriented system is used by end-users. End-users do not know the technology of service testing. The end-users do not have any scope for service testing, the only concern is that the system they use work correctly [14, 17].

B. SOA Testing Levels

We can explain different types of testing in SOA based application are as follows:-

a) **Unit Testing:** Unit testing is also known as service-component-level testing. Developer performed the unit testing to ensure that component functionality within the services work correctly. In unit testing a smallest piece of software is tested. This smaller piece is isolated from the remainder of the code. Now, the developer checks the behavior of this piece and to ensure it behaves exactly as we expected or not. Before service integration each component must be tested [35].

b) **Integration Testing:** In this testing phase our focus is on service interfaces. Integration testing ensures the interface behavior and sharing information between the services as specified. This test phase may include testing external services to organizations [35, 36].

c) **System Testing:** System testing test the SOA technical solution and to ensure business requirement and defined business acceptance criteria that have to be met. In this testing phase there is a need to understand the quality and test coverage of the previous test phases and this ensure the testing of key business scenario [35].

d) **Regression Testing:** Regression testing of an SOA system ensures that no previously working services fail as a result of modification or repairs. This testing phase determines the service up to point of repair has not been affected by the fixes. So, Regression testing is also known as validation testing [36].

Regression testing is important for web services due to its rapid developing speed and continuously changing in user demand. Normally *Rapid Application Development Model* (RAD) is used in the development of web applications that can adjust frequent changes of user demand and fulfillment of this change make the regression testing more important. When changes happen in the web services we have need to retest all the existing functionality to ensure that the changes are correct any new bug are not there and all the previous functionality is working correctly and at the same time we have to generate new test cases that are needed to test modified functions.

e) SLA Testing: SLA tests help us to identify the condition due to which service functionality can't be performed with a desired level of SLA. Before offering the service, the service provider would limit the service violation possibilities that can be violated during service usage. Service violations occur due to the combination of different factors, named: inputs, binding between abstract and concrete services, network configuration and server load [1].

SLA is a part of contracts between the provider and the consumer to spell out who is responsible for what. SLA contains information about each party and explain what they should do and what they should not to do. There is a commitment in SLA that ensure quality levels required by users and service providers so that they can interact effectively. *Web service level agreement* (WSLA) and web service agreement specification are the specifications that relate to SLA for SOA environment [33].

C. SOA Testing Tools:

Various tools used to test SOA based applications are- ANTS, J-Blitz, E-Test, SOAP Scope, SOAPUI, SOA Test, Web Service Tester, Service Integration Testing Tool (SITT), TAXI.

D. SOA Testing Challenges:

Various challenges to test SOA based services are as follows [4,11,12,15,17,18,19]:-

a) SOA is distributed in nature and cover different deployment configurations that makes SOA testing challenging.

b) SOA is dynamic in nature. It means service that use SOA is having adaptive in nature such as: services can add new service, integrate new services and remove old one services. Thus, this dynamic nature in SOA makes SOA regression testing more difficult.

c) Testers must have a knowledge of a combination of white box, black box and gray box testing. So, it is very difficult for traditional testers to test SOA based application.

d) There is unavailability of service structure and source code for services. It makes difficult to test SOA based application.

e) It is very difficult to determine a service workload parameter at *service level agreement* (SLA). This makes non-functional testing difficult.

f) Web-services may involve outside service providers who charge their service and make testing costly.

g) In SOA testing, testing environment is similar to the deployment environment that makes SOA testing costly.

h) SOA is heterogeneous in nature. In SOA, there are incompatible interfaces, incompatible technologies, platform and programming language. So, it requires various types of test engine.

E. Traditional Testing and SOA Testing

The testing of SOA based applications is completely different from the testing of traditional approaches. There are some difference between the traditional testing and SOA testing which is shown in Table 4. :-

Table. 4 Difference between traditional testing and SOA testing.

S. No.	Traditional Testing	SOA Testing
1.	A Major portion of the test is UI based testing.	There is no user interface (UI).
2.	Single agent testing, having unified group.	Multi agent testing, having different groups.
3.	There are static unit testing and integration testing.	Support dynamic testing. Testing may be done at run time when the service is going to evaluate.
4.	Centralized testing	Distributed testing.
5.	Testing activities mainly focus on functional testing and performance testing.	The scope of this testing is much more beyond the functional testing and performance testing. e.g.- SLA testing.

III. Overview of SOA Based Service Testing

Ofutt et al. [6] present new approach to test web services based on data perturbation. Data perturbation is a process to modify request message, resending the modified request message and analyze the response message for correct behavior. Here, data perturbation is used to test peer-to-peer interaction between service. Two methods are introduced related to data perturbation: data value perturbation and interaction perturbation. This approach is applicable only between two components at a time. Tsai et al. [30] compare traditional software testing and web service testing and propose *web service group testing* (WSGT) technique to test composite services. *Web service group testing* (WSGT) can be used to rank the web service. The web services may be either unit or composite web services. We can use WSGT at each level of web service testing. In WSGT, there is an oracle for each input generated by a voting service mechanism. This oracle can generate automatically based on the majority principle. Bai et al. [22] generate test cases of individual service automatically based on the WSDL. WSDL carries the data transmission information and interface operation information about a service. Here, test data generation and test operation generation are two perspectives to generate test cases. Generation of test data is done by analyzing the message data types. We can analyze operation dependency and generate test operation. Tsai et al. [20] propose several testability evolution criteria to test SOA based software. To evaluate the support to test SOA software these evaluation criteria serve as a reference for both service providers and application builders.

Zhang et al. [25] extends UML 2.0 activity diagram to describe the syntax and behaviors of BPEL. This paper map BPEL primitive activity to UML 2.0 activity diagram. Next, mapping done from *business process execution language* (BPEL) structure activity to UML 2.0 activity diagram. This paper discusses the feasibility of testing using web service business process testing. To generate XML instances from the XML schema *XML based partition testing* (XPT) approach [3] has been discussed and *Testing by Automatically Generated XML Instances* (TAXI) tool has been implemented. With the help of an XML partitioning testing approach we can divide the input domain into a sub - domain such that, within each of them there is no change in the program behavior. Instances can be invalid because it is used to verify the robustness of the tested applications. XPT analyzes given XML schema file to simplify its partitioning. This process in XPT is called preprocessor. Preprocessor changes the syntax by making some transformation but preserve the meaning of XML schema elements. Yuan et al. [16] proposed the model driven approach for generating the executable test cases from the given business process. This approach has three stages:

1. Defining a *process under test* (PUT) based on the business process model.
2. Transforming the process under test to abstract test cases and
3. Transforming abstract test case to executable test cases.

The business process model is composite of activity, control node, client and web service. *Process under test* (PUT) is visualized in activity diagram and *abstract test case* (ATC) is visualized in sequence diagram. Khan et al. [8] proposed a model-based approach that support regression testing to isolate the changes. It helps to retest the affected parts. The model describes the interface of services before and after changes and able to identify the changes and their impact. Jiang et al. [24] present a method for test data generation of web services. Generation of initial test data is done from the WSDL description information and contract then selection of test data is done through contract mutation testing. WSDL defines a semantic information including pre-condition and post-condition for the interface of a web service. Precondition expresses the condition under which interface of a web service functions properly. Post-condition express execution result when a web service interface has been executed correctly. Naslavsky et al. [9] give an approach that creates a traceability relationship between model elements and test cases during test case generation which are used to support model based regression test selection. Test cases are derived from a models describe expected result and expected behavior. Here, test cases are introduced as an obsolete test case, retest able test case and reusable test case. The obsolete test case is no longer valid. Test cases are deleted. Test cases that cover modified code come into the category of retestable test cases and those that has not been modified come into the category of reusable test cases. Reusable test cases are not used for regression testing [37].

Test case prioritization strategies using XML message sequencing is introduced in [2] to reorder regression test cases for composite web services against the tag based techniques. If the test case includes an XML message that contains an XML element we say that XML element is covered by the test case. To cover various services test cases having a particular sequence. Chen et al. [27] used an activity diagram and proposed an approach to generate executable test cases automatically. Here, they use simple path coverage adequacy criteria. There are initial node and final node that represents the start and end of the activity respectively. This simple path represents a feasible execution in an activity diagram. Before, the execution of randomly generated test cases, there is a need to calculate decision nodes of a data classifier for predicting the execution path of activity diagram. Inputs that are unable to identify the path coverage will be dropped. To generate the test cases from activity diagram, Boghdady et al. [13] proposed an enhanced XML- based automated approach. To cover the functionality of the activity diagram an *activity dependency table* (ADT) is created for each XML file. There is a precondition and postcondition in the activity dependency for carrying the input, output and guard conditions of the removed nodes in the form of expression. Activity dependency graph is generated from *activity dependency table* (ADT) to show all the possible test paths. Test cases derived from behavior instructional models are functional test case and they have the same level of abstraction as the model creating them. Full condition coverage criteria used with state chart or communication diagram. All basic path coverage criteria used with activity diagram.

Tsai et al. [26] introduced a verification mechanism to the UDDI server including check-in and checkout of web service. Check-in means web service pass through various test before they can be registered in the UDDI. Checkout process allows a client to test drive web service before it releases to the client. The test script should be attached to WS and used by both web service providers and clients. Test monitor, test master and test agent make the test infrastructure for the UDDI server to test web services remotely. Mei et al. [10] proposed a novel testing approach named *preemptive regression testing* (PRT). PRT preempts the execution whenever a late change in the service under regression test is detected. PRT now selects test cases from a regression test suite as fixes, run the fixes and then resumes the suspended execution of the regression test suite. This process is repeated until there is no test execution preemption between any two test cases of the whole test suite occurs. Tsai et al. [31] introduced an XML based object-oriented testing framework 'Coyeto' to test web services. Here, Coyeto focus on integration testing. Coyeto having two parts named: test master and test engine. Test script is generated by test master and configuring, testing, validating and logging of this test script is done by the test engine to generate test result. Method name and input-output parameter is described in the WSDL document. Test master generates test scenario by extracting interface information of WSDL. Test cases are generated in the XML format from this test scenario which is interpreted by test engine.

LI et al. [32] proposed BPEL4WS unit testing framework and introduced the challenges of BPEL process testing. In business process unit testing test cases are design from internal logic. Here, BPEL interact with the partner process. Partner process participates two way interaction with the process by providing services to the process and require service from the process. BPEL process composition model. Test architecture, Lifecycle management and test behavior design are the four parts of BPEL framework. Kuppuraju et al. [23] propose a methodology to verify the interoperability of the various products in an SOA stack. This paper also discussed the interoperability problems and their solutions. The interoperability problem arises due to many reasons such as: - differences in the version of web service standard and specification, differences in the protocol support and differences in the error handling mechanism. Interoperability problem is solvable by many ways such as: - by identifying service and interface, identifying integrated part of the SOA stack and by identifying use case and non-functional requirement. Athira et al.[29] proposed test case prioritization technique using activity diagram and identifies the differences between the original model and modify the model. Based on the model information activity path is plotted. The test case that covers these activity path are the most beneficial test cases. In these activity path affected path is the path that is affected by the addition or deletion of transition during the execution of the modified model.

Atkinson et al. [21] introduced semantics of test sheets and propose a new approach to test service which combine the expressive power of framework integration test (FIT) and testing and test control notation. Input test sheet and result test sheet are the two basic forms of the test sheet. Input value and result of the system under test are defined in the input test sheet. The result of the test execution is described by the test result sheet. Result sheet contains all the information of input test sheet and additional information about the system under test. Each row of the test sheet represents a call to the system functionality. Operation invocation with different number of parameters is represented by different raw of the test sheet. Actual value or return value is displayed by result test sheet. Loini et al. [28] proposed a framework for service Integrator to collaborate during service testing by making a test suite open to the public and share testing result. When we call a web service we delegate part of our business to an external service provider doing it for us. There is no control on the execution part of the business logic. The role of service Integrator has identified the potential service, verify that service does what they are suppose to do, formulate SLA with a service provider and compare the business need with the offered service. Service integrators are able to write their test suite based on providing service specification. These test suites are open to the community.

IV. SUMMARIZED

Following is the year wise descriptions of various SOA testing proposed work:

SOA Testing Proposed Works

A. The contributions of year 2002 [31]

Web Service is tested using XML based object-oriented testing framework Coyeto. Coyeto having two parts a. Test Master and b. Test Engine. Test master generates test scripts to test the engine and test engine configuring, testing, validating and logging test script and generate test result. WSDL contains a method name and input-output parameter. Test master generates test scenario by extracting interface information of WSDL. The test case is generated from the test scenario in the XML format which is interpreted by test engine. Test Engine reads the test script provided by test master and executes the test at the target web service and send the test result back to the test master. Here instead of module testing Coyeto focus on Integration testing [31]. But, as the importance of regression testing in web services increases there is more need to focus on it.

B. The contributions of year 2003 [42]

Introduced the slicing approach for web application regression testing. Slicing method divides a problem into blocks and we can solve a relevant block with the concentrated strength. This paper introduces a concept of in-degree and out-degree. The number of hyperlinks in one page is called its out-degree. In-degree of a page is defined as the number of all the hyperlinks which point to one page. If the in-degree of a page is zero then that page is considered as isolated page. The pages which is pointed by hyperlink add 1 to that page. In this paper they divided the relationship between the related pages in two ways : direct and indirect dependent. For direct dependent relationship testing we only need to validate whether these pages are reachable and effective. For indirect relationship we need to validate data transfer and share variable [42]. Here, optimization and prioritization of test cases for identifying the dependency remains.

C. The contributions of year 2004 [30]

The component can be dynamically replaced during execution if the component fails in the composite service. This paper focus on collaborative verification and validation instead of traditional independent verification and validation. Comparison between traditional independent verification and validation and service-oriented collaborative verification and validation is also introduced. This paper proposes a WS group testing technique for testing a composite service and also explain how blood group testing is different from web service group testing [30]. There is a need to address the aspect to identify the faulty participating web services into a composite web service and verifying group testing by voting at regression levels also need to address.

D. The contributions of year 2005 [22] [32]

Web service testing is automatically based on WSDL web service specification language. WSDL contains information about the service interface, Service operation, and the data that is to be transmitted. There is multiple

operation in a service. We generate a test case for testing individual operation as well as a sequence of operation. The generated test cases are encoded in XML called service test specification. This paper introduces the web service automation testing technique based on WSDL specification. This paper mainly focuses on the atomic service automation testing and for this operation dependency analysis are used. Here three types of dependency are identified: - Input dependency, Output dependency and input/output dependency and these dependencies are used in test generation. This paper presents the algorithm of automatic test case generation for distributed service testing [22].

The authors of this paper himself describe the problem which is needed to be solved are:

a. Service flow specification analysis and automatic test case generation for composite services.
b. Semantic service specification analysis to explore more semantic information for intelligent test case generations.

BPEL interaction with the partner process. Partner process may provide service to the process, require service from the process and thus participate two way interaction with the process. This paper also introduces challenges for BPEL process testing. In business process unit testing internal logic are known and based on the internal logic test case are designed and these test case help to test the logic thoroughly. In this paper BPEL4WS unit test framework is introduced. BPEL framework having four parts [32] -

a. BPEL process composition model
b. Test Architecture
c. Lifecycle management
d. Test behavior design.

Here, author itself indicates the need to automate test generation from BPEL source code that aligns with the proposed test framework.

E. The contributions of year 2006 [12] [20]

A regression testing approach introduces that can be performed on a UML model. A new test case is generated when the when the selective retest suit does not cover all the changes. This paper focus on selective retest strategies. Selective test case are those that cover code that has been modified and thus these test cases are used for regression testing. Selective retest strategies are efficient if its runtime is less than a retest -all strategies. If the runtime is equal to the retest-all strategies than it is said to be safe. This paper also introduces the issue rise in UML based regression testing. These issues are :-

a) Classifying and identifying changes made for different version of UML design.
b) Select test case for regression testing to ensure safety and efficient use of the changes.
c) Determine whether new test can be added.

Instead covering the behavior of the code this paper cover only UML and interested to identify the changes that affect UML test cases. In this paper OMDAG (object method directed acyclic graph) concept is to also introduce. OMDAG map the dynamic information in a sequence diagram to a direct acyclic graph. Mapping associate method in the sequence diagram with their originating objects. Edge of OMDAG represents traversing successive method execution. Combining OMDAG with the class tuples information. Here class tuples consist of the class name, attribute and methods. Now OMDAG represent the integrated model that combines class diagram and sequence diagram. Execution of test case traverses the path in the OMDAG [12].

There is more need to describe the way to reduce the test suite complexity. Because if the integrated module increases than the number of UML view also increases and it increases the complexity of test suites. This makes regression testing more important.

This paper proposes service testability evaluation criteria. Service evaluation criteria serve as a reference for both service provider and an application builder to evaluate the test support of SOA software. This paper introduces the concept of DCP and compare it with traditional independent verification and validation and service-oriented collaborative verification and validation. We need to re-specify the workflow when the collaboration is done at run time. It makes DCP more challenging. At DCP actual collaborations are unknown until the services dynamically established the protocol. The typical DCP process is divided into four stages-

a) Preparation / Profiling stage- At this stage CPP which contains a list of protocols can be updated as a service terms from actual collaboration.
b) Establishing Stage- Participating Service exchange their CPP (Collaboration protocol profile) and agree on common protocol CPA (Common protocol agreement) that all participating services share.
c) Execution Stage- In this phase earlier CPA participating service will collaborate. Data may be collected so that CPP can be updated for future collaboration.
d) Termination Stage- Participant service terminate the collaboration and update their own CPP based on data collected.

In DCP dynamic configuration and collaboration are established at runtime , verification needed to be performed at run time. Test coverage can be completed when the actual collaboration is established [20].

Here, authors need to identify fault participating web service during dynamic composition, dynamic re-composition and dynamic service collaboration.

F. The contributions of year 2007 [23]

This paper built a methodology to verify the interoperability of the various products in an SOA stack. In SOA stack various products are consisted such as: - Business process management (BPM), Enterprise service bus (ESB), Service Registry etc. This paper also focus on interoperability problem and their solution. The interoperability problem arises due to the various reasons:-

a) Differences in the version of web service standard and specification.
b) Differences in the error handling mechanism.
c) Differences in the protocol support.

These interoperability problems are solvable by the following way:-

a) Identifying integration part of the SOA stack.
b) Identifying use case and non-functional requirement.
c) Identify the service and interface.

This paper little introduce some testing for ensuring interoperability. For example- Standard compilation test is executed against the service using the WS-my compilation tool. The SOA integration test covers all the positive and negative scenario of identifying business processes. Integration testing test business exception and system generated exceptions. In SOA performance test load and longevity test covered. The message size test has also executed to check the stability of SOAP message on SOA stack [23].

There is more explanation needed to cover interoperability testing. A tool is needed to generate a report of the interoperability issue.

G. The contributions of year 2008 [43] [44] [45]

Model based regression test selection technique is based on the design model. This paper proposed a concept called behavioural slicing to structure the activity diagram. Use case activity diagram (UCAD) represents the functional behaviour of the given use case. UCAD is a requirement specification model that is developed during the requirement analysis phase. This approach also generates new test case including modified test case to validate changed software. This paper proposes a novel concept called behavioral slicing to provide structure to the activity diagram. When we consider behavior slicing each use case is considered into a set of unit of behavior. Unit of behavior is a tuple consisting of system processing, system output, optional candidates and user action. Here each unit of behavior represents a user action followed by system processing and system output. This approach identifies the impact on software when there are changes made to the specification. Based on this impact a test case to be re-executed to validate the software being modified. System behavior is modelled through the UCAD. Each use case / user scenario contains the system behavior. In activity diagram each node is identified using a unique node version and with this unique node version we can easily identify the changes made to the activity diagram, due to the changes made to the software requirement. These changes are classified as:-

- Modification to the existing node
- Addition of a new node
- Deletion of a new node
- Shifting(Deletion followed by addition) of an existing node.

Regression test suit to be selected and re-executed [43].

When we see this approach we found that in this approach person need to keep track of node version when creating an activity diagram because each node version has to be manually typed with the description at the time the node is created.

This paper presents a new regression testing approach for web applications that covers changed element and other potentially affected one. This paper proposes a safe regression testing technique based on event dependency graph and introduced slicing based regression testing technique for web application. Slicing give tester more focus on simplified contents and improve the working efficiency. This paper used to event dependency graph (EDG). First EDG is used to model the original system and another EDG is used to represent the modified system. Event based dependence is classified into three types:-

- Link Dependence: - It is usually the hyperlink. It holds between two pages if the first request the second.
- Visible Effect Dependence: - It also exists between two pages where the requesting page modified the another page and that another page open with the modified data.
- Invisible Effect Dependence: - When one page modified the other without displaying the effect.

If G represents the original web application and G' represent update web application then we have to identify the change node and the potentially affected node. Here potentially affected node are the nodes that are connected directly or indirectly to the affected nodes. All the dependence in the graph is taken into consideration to identify the affected nodes. For each affected node goes to the test suit T and check if the node is present in the test cases. It the node is present in the test cases add the test case to the new test suit T' [44].

There is a need to generate test sequences of modified and potentially modified components.

Services are designed to function as independent entities may work perfectly well in isolation but when we integrate the service into the application might not function as expected. Services need to be tested for data translation and information delivery for different consumers. This paper discusses overall SOA testing methodology and additional test area specific to SOA. There is reusability, agility, interoperability and security based testing before they are consumed by composite application. In this paper testing is done as a test engineer basis. This paper introduces some of the test area for SOA testing:-

a) Service Agility Testing: - This testing is related to the configuration, Business rules and policies.
b) BPEL Level 1 Testing: - This type of testing ensures compensating transaction and service unavailability impact.
c) BPEL Level 2 Testing: - This type of testing area covered workflow testing and testing event.
d) Security Testing: - Here denial-of-service, vulnerability, context propagation/ federation are tested
e) Service Design Testing: - Here interoperability testing, service-app integration testing, reuse testing and service data testing has become [45]. Reuse and required agility at the enterprise level need to be tested.

H. The contributions of year 2009 [8] [9] [24]

This paper proposes a model based regression testing approach. A model describing service interfaces before

and after the changes are compared in order to analyze the system evolution and identify which test need to be rerun and where new one are required. This paper describes the model which specifies the external behavior and external data flow of the service. A trace can be presented by sequence diagram. Sequence diagram describing the message flow between different actor in the system. If M is the trace before and M' Is the trace after the evolution of the system than we derive three subsets A, B and C such that:-

A= M∩M' are the trace that are preserved.

B= M'/M are completely new trace.

C= M/M' are the changed traces containing the invocation of old as well as new methods.

Data dependence can be visualized by bipartite dependence graph over nodes where oval representing methods and square representing classes.

For selection of test cases from A three subset A1, A2 and A3 are derived, two subsets are derived from C named C1 and C2 and B contains new functionality so there is no any subset of B. A1 having no external or internal dependencies with changed part of the system , A2 having an internal data modification model and A3 having changed with the implementation of the method. C1 has addition/deletion/ modification operation with the internal data model as well as external modification as in C1.

For dependencies analysis author used a graph transformation system. There is a number of transformation rule in the graph transformation system where the system state before operation is represented by left- hand -side and system state after operation is represented by the right-hand-side [8].

When we take a model based approach to our testing process be in a higher abstraction level. In model based testing there is a relationship between test case and model element. Model based testing test what the code is supposed to do and thus complementing code based testing which test what the code does. In model based testing we can continue with certain testing activity when the code is unavailable. This paper proposes a fine-grained traceability relationship between the model element and test cases that traverse those elements and locate test cases for retest and used to support model based regression test selection. This paper use UML class and sequence diagram and its modelling perspective [9].

This paper present automated test-data generation method for web services based on a contract based mutation testing techniques. Test suite generates by this method which indicate the quality and efficiency of testing. This paper introduces three sources of test data:-

a) Function Based: - Which are gained from software requirement.
b) Structure Based: - Which are gained from the implement.
c) Error Based- This is based on typical and useful errors during the development process.

In web service contract design-by-contract is a method to improve software testability. It establishes the contract between the provider and the user of software entity during testing. If execution brakes one or more contracts than it can be viewed as a fault revealing execution and broken contract can be used to trace the fault.

Typical contract includes precondition, post condition, class invariants, loop invariants. Precondition expresses the condition under which interface of web services functions properly. Post condition expresses the condition under which web service interface executed correctly.

Contract fault model listed as contract negation, condition exchange, precondition weakening, post condition strengthening [24].

I. The contributions of year 2010 [29] [2] [21]

This paper proposes a model based test prioritization using activity diagram and identifies the difference between the original model and modified model and based on this information plot an activity path for each test case and identifies the most promising path. The test case that cover this path are the most beneficial test cases. In activity diagram node represents activity node and link represent transitions between two activities. Affected path is the path that is affected by the addition or deletion of transition during the execution of the modified model on test T' [29].

Next, we have to perform an experimental study on larger models and systems for better understanding of model-based testing prioritization.

This paper proposes new test case prioritization strategies using XML message sequencing to reorder regression test cases for composite web service against the tag based techniques and disclose how the test case uses the composite service interface specification. The XML file is parsed to check whether the input is valid or not. If there is a valid input then write result in the output file and proceed as par the activity diagram else exist from the sequence of invocation. Member services are constantly involved in service composition. Other services that relate to this member service need to conduct regression testing to verify the function of the latter services and also conform interoperability [2].

There is a need to enhance the propose technique by integrating other prioritization index such as: - cost, resource based etc.

This paper proposes a new approach of service testing which combine the expressive power of framework integration test (FIT) and testing and test control notation. This paper introduces a semantics of test sheets and show some high level feature offered by the test sheet framework.

Test sheet having two basic forms input test sheet and result test sheet. Input test sheet defines the input value and the result that the system under test is expected to return. Result test sheet on the other hand describe the result of a test execution . Result test sheet contains all the information of an input test sheet and the additional information about the reaction of a system under test to the stimulus. Each row of a test sheet represent call to the system's function. In the test sheet input parameter and output parameter is always separated by two distinct areas separated by double line called as invocation line. Different raw in a test sheet represent the invocation of

operation with different number of parameters. The number of columns in input area is determined by highest number of parameters of the operation to be tested. In the output column the given value is not the actual value but represent the expected result defined by a tester. The actual value or return value is displayed by the result test sheet [21].

We need to extend the test sheet to measure timing , deviation and other key characteristics of web services. This measure can be combined with a service registry in order to create a QoS (Quality of Service) aware service repository.

J. The contributions of year 2011 [28] [46]

When we call a web service we delegate part of our business logic to an external provider doing it for us and we have no control what could happen during the execution of that part of the business logic. Many testing approaches have been proposed to address various aspects related to web service testing.. This paper proposes a framework for service Integrator to collaborate during web service testing by making a test suite open to the public and share testing result. The role of service Integrator is:-

a) Identifies the potential services
b) Compare the business needs of their system with the offered service.
c) Verify that the service does what they are suppose to do.
d) Formulate the service level agreement with the service provider before starting using the services function.

This paper assumes that the web service is meant to be used by service Integrator and they have a minimum knowledge about the testing. They are able to write their test suites based on providing specification and these test suites are open to the community and can be shared among them. The community member rate the test suites to identify most useful scenarios used by the each web service. Test suite management, feedback management and user interface are the three blocks that make the test suite server. Test case management performed following six consecutive steps [28]:-

a) Storing test suites
b) Data extraction
c) Test data generation
d) Test case generation
e) Test case execution
f) Result validation

Further, we need to use this proposed framework on public services and dataset releases for traces.

This paper do the implementation of the work presented in its earlier paper [28]. To implement a test suite server we have used some existing components to perform different tasks. TestGen4J is a collection of open source tools that automatically generates unit test cases. WSDL4J allows manipulation of WSDL documents. SOAPUI used for service invocation and capturing the response messages. SOAPUI provide much more functionality including WS-mocking and load testing. A simple JSP page handles the user ratings and store the received rates into a related database and this rating rank the test suite [46].

K. The contributions of year 2012 [48] [17] [47] [4]

Service -oriented architecture remove the gap between software and business. This paper reviews SOA testing challenges and existing solutions for those challenges. This paper introduces functional testing challenges and a non-functional testing challenges. On unit testing service do not have the service interface so the testing team must have enough skill to produce good test, test object and required test data. Service unit testing differences in input-output type with component testing and this makes more complex to generate test data. Availability, Performance, Applicability, Maintenance capability and portability are the non-functional properties of the system. It is very difficult to determine a service workload parameter at a service level agreement for non-functional testing. To overcome the challenges of functional testing a tool is used for performing a complex action. Use ESB capabilities for the functional testing system, Other technologies such as JMS middleware was used. TASSA framework is used for automatic testing in functional and non-functional specification of service based application. Another tool WSOTF presented for the automated testing [48].

Regression Testing of SOA based applications conducted during the maintenance phase may present several challenges. This paper explores a roadmap to regression testing of SOA based applications. The SOA based application is a combination of web components, middle-tier components, server and legacy system. A modular application software can be classified into three categories:-

a) Component based software
b) Reuse oriented software
c) SOA-based software

In a component based software development separation of concerns on wide-ranging functionality available throughout a given software system. Software engineers regard components as part of the starting platform for service-orientation. In web service or in SOA component is converted by the web service into service.

In reuse-oriented software reuse of software is incorporated. This is usually happens when people working on the project know of design or code which is similar to that required, modified them as required and incorporate them into a system. SOA represents an architectural style that incorporates service orientation. This paper introduces a testing perspective as service developer, service provider, service Integrator, third-party certifier and end user. Different testing level such as unit testing, integration testing, system testing, regression testing is introduced shortly. Regression testing in SOA based application represents selective retesting of components so that modification does not affect other components as well as the system as a whole. Unlike unit testing, Integration testing or system testing which is performed software Lifecycle, regression testing is performed during the maintenance phase of SOA based application [17].

This paper done a survey work for regression testing method for SOA. The changed scenario of implementation of SOA enable application has been considered. This paper introduced a comparison between various regression testing approach named selective regression testing, code based regression testing, risk based regression testing, state based regression testing and model based regression testing [47].

This paper proposes a new automated, distributed, cross-platform and regression testing architecture for testing SOA application and their web service components. The propose frame work is made out of test engine units capable of conducting regression testing. Test engine unit co-ordinate, control and manages various testing units and their processes. Test code generator unit generating test script and client code necessary for test execution. Test case generator unit generating test condition, variables and data sequence and all test scenarios. A test executor unit applying test case to the web service. Test monitor unit evaluating the testing result, database that store valuable testing parameter throughout the testing process [4].

Non-functional aspects of SOA applications and bring it in a complete testing solution that can not only test SOA functional operation but also non-functional qualities such as- Performance, Security, availability and scalability are missing.

L. The contributions of year 2013 [49] [50]

Jokhio et al. [49] proposed a goal oriented generation and mutation based evaluation approach towards service testing. The goal specification of a web service contains crucial information in terms of the pre-conditions and effects of the behaviors of the expected functionalities, which could be used for testing purpose. The mutation score is a ratio of faults discovered versus the total number of the faults injected into the program. Number of faults injected is returned by calculating the total number of mutants generated. Each generated mutant represents a single atomic fault that is injected in the web service implementation. Number of faults discovered is determined based on the mutant execution and the comparison results of the two dimensional array. This is done by counting the number of rows that have at least one occurrence of 1 in a particular row, which indicates that the particular mutant is detected by some test case. Number of faults undiscovered are calculated by counting the number of rows in the results array returned by mutant execution which do not have any occurrence of 1. This indicates that the particular mutant is not detected by any test case. It is important to perform further analysis of why such mutants were not killed by any test case.

Bhuyan et al. [50] proposed regression testing process and SOA testing perspective models. Regression testing process is defined with the help of UML use case diagram and activity diagram. UML use case diagram helps us to break our requirements into short stories that makes easy to understand. Use Cases focus on the user of the system and describe the way the system can be used by the user. In activity diagram nodes represent various user actions, conditions and system outputs. The edges represent transitions from node to node.

Authors also proposed SOA testing perspective models (STPM). There are three different perspectives of this model. STPM represents a complex model consisting of Service Developer Model (SDM), Service Tester Model (STM) and Service Provider Model (SPM). In the sub-model SDM Service Developers know the internal structure of the service. They have a knowledge of service specification. Service Developers test the services in terms of service functionality, ensure quality of services and the interaction with other services. Service Developers deliver both interface and implementation of the services and are responsible for detecting bugs in order to release reliable services. In the sub-model STM Service Testers execute and manage the test cases. Service testers first identify the changes in the services. After identification of changes, the tester needs to identify the missed coverage items. The tester needs to select and execute test cases that cover missed coverage items in the service. The execution order of the test cases should be in such a way that the test cases that cover more number of items should be executed first. In the SPM Service Provider publishes services in UDDI. But before publishing, it is necessary to test a service so that the service provider can be able to give guarantees about the quality of the registered services. The service provider generates test cases from a model provided by the service developer. The service provider submits test cases to the UDDI to confirm service standards. During registration of a service, the server will use the

provided test cases to test the services. The service provider also sets various criteria and matching rules for service consumers and ensure that the authorized service consumers are able to use this service. There is also a need to develop test cases for notification mechanism. This ensures that, when there is any evolution in the service or any new version of the service is registered in the infrastructure, an automatic notification is sent to the service consumers. The changes happening in the UDDI registry is done by the authenticated users. unauthenticated users can also access the UDDI registry for read only purposes.

V. CONCLUSION

This paper presented various testing perspectives and various testing challenges of SOA based services. This paper also discussed many existing approaches that addressed the problems of SOA testing and improve the testability of SOA based application. Further, we also discussed SOA and web service differences, traditional testing and SOA testing differences. This paper also identified the need to generate automatic test cases with semantic service specification analysis. Another aspect of this paper is to identify various testing problems presented in different testing approaches.

ACKNOWLEDGMENT

We express our greatest gratitude and appreciation to school of computer engineering, KIIT University, Bhubaneswar, Orissa, India.

REFERENCES

[1] Gerardo Canfora. "Service-Oriented Architectures Testing: A Survey", Lecture Notes in Computer Science, 2009.

[2] Askarunisa, A., A. M. Abirami, K. Arockia Jackulin Punitha, B. Karthik Selvakumar, and R. Arun kumar. "Sequence-based techniques for black-box test case prioritization for composite service testing", 2010 IEEE International Conference on Computational Intelligence and Computing Research, 2010.

[3] Andrea Polini. "Automatic Test Data Generation for XML Schema-based Partition Testing", Second International Workshop on Automation of Software Test (AST 07), 05/2007.

[4] Youssef Bassil. Distributed, cross-platform and regression testing architecture for service-oriented architecture. In *proceedings ofAdvance in computer science and its applications(ACSA)* (2012), vol. 1.

[5] http://www.sysed.com/tech_assessments/leader/web_serversasp?source=http://www.databasetrainingbysysed.us.

[6] Jeff Offutt and WuzhiXu. Generating Test Cases for Web Services Using Data Perturbation. In *IEEE* (2003).

[7] Jeffrey Hasan and Mauricio Duran. *Expert Service-Oriented Architecture in C# 2005*. Apress, 2006.

[8] Khan, T. A., and Heckel, R. A Methodology for Model-Based Regression Testing of Web Service. In *proceedings of Testing: Academic and Industrial Conference-Practice and Research Techniques, IEEE* (2009).

[9] Leila Naslavsky. " A model-based regression test selection technique", 2009 IEEE International Conference on Software Maintenance, 09/2009.

[10] Mei, Lijun, KeZhai, Bo Jiang, W. K. Chan, and T. H. Tse. "Preemptive Regression Test Scheduling Strategies: A New Testing Approach to Thriving on the Volatile Service Environments", 2012 IEEE 36th Annual Computer Software and Applications Conference, 2012.

[11] Massimiliano Di Penta, Marcello Bruno and Gerardo Canfora. Web service Regression Testing. In *RCOST-Research centre on software technology* (2007).

[12] OrestPilskalns, GunayUyan and Anneliese Andrews. Regression Testing UML Design. In *Proceedings of the 22nd IEEE international conference on software maintenance* (2006).

[13] Pakinam N. Boghdady. "An enhanced test case generation technique based on activity diagrams", The 2011 International Conference on Computer Engineering & Systems, 11/2011.

[14] Kalamegam, Poonkavithai and Godandapani, Zayaraz. "A Survey on Testing SOA Built using Web Services", International Journal of Software Engineering & Its Applications, 2012.

[15] PrachetBhuyan and Abhishek Kumar. Model Based Regression Testing Approach of Service-Oriented Architecture (SOA) Based Application: A Case Study. In *International Journal of Computer Science and Informatics* (2013), vol. 3.

[16] Qiulu Yuan. "A model driven approach toward business process test case generation", 2008 10th International Symposium on Web Site Evolution, 10/2008.

[17] Rajanikanta Mohanty, Binod Kumar Pattanayak, Bhagabat Puthal and Durgaprasad Mohapatra. A Road Map to Regression Testing of Service-Oriented Architecture(SOA) Based Applications. In *Journal of theoretical and applied information technology* (2012), vol. 36.

[18] http://arxiv.org/ftp/arxiv/papers/1203/1203.5403.pdf.

[19] Yongbo Wang. "Business Semantics Centric Reliability Testing for Web Services in BPEL", 2010 6th World Congress on Services, 07/2010.

[20] W.T. Tsai, Jerry Gao, Xiao Wei and Yinong Chen.Testability of Software in Service-Oriented Architecture. In *Proceedings of the 30th Annual International Computer Software and Applications Conference, IEEE* (2006).

[21] Atkinson, Colin, Florian Barth, Daniel Brenner, and Marcus Schumacher. "Testing Web-Services Using Test Sheets", 2010 Fifth International Conference on Software Engineering Advances, 2010.

[22] XiaoyingBai and Wenli Dong. WSDL-Based Automatic Test Case Generation for Web Services Testing. In *Proceedings of the 2005 IEEE International Workshop on Service-Oriented System Engineering, IEEE* (2005).

[23] SujathaKuppuraju,Aravind Kumar, GeethaPresennaKumari, "Case Study to Verify the Interoperability of a Service Oriented Architecture Stack", IEEE International Conference on Services Computing (SCC 2007), IEEE, 2007.

[24] Ying Jiang, Ying-Na Li, Shan-Shan Hou and Lu Zhang. Test Data Generation for Web Services Based on Contract Mutation. In *Proceedings of Third IEEE International Conference on Secure Software Integration and Reliability Improvement, IEEE* (2009).

[25] ZhangGuangquan, Rong Mei and Zhang Jun. A Business Process of Web Services Testing Method Based on UML 2.0 Activity Diagram. In*Proceedings of Workshop on Intelligent Information Technology Application, IEEE* (2007).

[26] W.T.Tsai, R.Paul, Z.Cao, L.yu, A.Saimi, B.Xiao, "Verification of web services using an enhanced UDDI server", In *the Proceedings of The Eighth IEEE International Workshop on Object-Oriented Real-Time Dependable System, IEEE,2003.*

[27] Chen, Xin, Nan Ye, Peng Jiang, Lei Bu, and Xuandong Li. "Feedback-Directed Test Case Generation Based on UML Activity Diagrams", 2011 Fifth International Conference on Secure Software Integration and Reliability Improvement - Companion, 2011.

[28] El Ioini, Nabil. "Web Services Open Test Suites", 2011 IEEE World Congress on Services, 2011.

[29] Athira B, Philip Samuel, "Web Services Regression Test Case Prioritization", In *proceedings of International Conference on Computer Information Systems and Industrial Management Applications (CISIM), IEEE*, 2010.

[30] W. T. Tsai, Y. Chen, R. Paul, N. Liao and H. Huang. Co-operative and Group Testing in Verification of Dynamic Composite Web Services. In *Proceedings of the 28th Annual International Computer Software and Applications Conference IEEE* (2004).

[31] W. T. Tsai, Ray Pau, Weiwei Song, Zhibin Cao, Coyote:" An XML-Based Framework for Web Services Testing", In *the Proceedings of 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02), IEEE* 2002.

[32] ZHANG, Zhongjie LI, Wei SUN, Zhong Bo JIANG, Xin ZHANG, "BPEL4WS Unit Testing: Framework and Implementation", *International Conference on Web Services (ICWS'05), IEEE*, 2005.

[33] http://www.scribd.com/doc/ 12863121/Service-Level Agreements-in –ServiceOriented –Architecture –Environments.
[34] http://www.altova.com/whitepapers/webservices.pdf.
[35] http://www.soatutorial.net/soa-test-approach-the-purpose-and-how-to-do-it/.
[36] http://www.thbs.com/pdfs/SOA_Test_Methodology.pdf.
[37] Vincent, Pierre-Luc; Badri, Linda and Badri, Mourad. "Regression Testing of Object-Oriented Software: Towards a Hybrid Technique", International Journal of Software Engineering & Its Applications,2013.
[38] SOA and Web Services, http://www.oracle.com/ technetwork /articles/javase/soa-142870.html.
[39] Difference Between SOA and Web Services, http://www.differencebetween.com/difference-between-soa-and-vs-web-services/.
[40] What's the difference between SOA and Web services,http://searchdatamanagement.techtarget.com/answer/Whats-the-difference-between-SOA-and-Web-services.
[41] Thomas Erl. Service-Oriented Architecture Concept technology and Design. Pearson Education, 2005.
[42] Lei Xu, Baowen Xu, Zhenqiang Chen, Jixiang Jiang, Huowang Chen," Regression Testing for Web Applications Based on Slicing", 27th Annual International Computer Software and Applications Conference (COMPSAC'03) 0730-3157/03, 2003 IEEE.
[43] Ravi Prakash Gorthi, Anjaneyulu Pasala, Kailash KP Chanduka and Benny Leong, " Specification-based Approach to Select Regression Test Suite to Validate Changed Software", 2008 15th Asia-Pacific Software Engineering Conference, IEEE, 2008.
[44] Abbas Tarhini, Zahi Ismail, Nashat Mansour, "Regression Testing Web Applications", International Conference on Advanced Computer Theory and Engineering, IEEE, 2008.
[45] Srikanth Inaganti and Sriram Aravamudan, " Testing a SOA Application" BP Trends , April 2008.
[46] Nabil El Ioini, Alberto Sillitti, "Open Web Services Testing",2011 IEEE World Congress on Services.
[47] Prachet Bhuyan, Chandra Prakash Kashyap, Durga Prasad Mohapatra," A Survey of Regression Testing in SOA, "International Journal of Computer Applications (0975 – 8887) Volume 44– No19, April 2012".
[48] Ebrahim Shamsoddin-Motlagh," A Survey of Service-Oriented Architecture Systems Testing", International Journal of Software Engineering & Applications (IJSEA), Vol.3, No.6, November 2012.
[49] M. Shaban Jokhio, Gillian Dobbie, J. S., and Hu, T. Web services testing via goal and mutation. In Proceedings of IEEE International Conference on Engineering of Complex Computer Systems (2013), pp. 159-162.
[50] Prachet Bhuyan, Abhishek Kumar and D.P. Mohapatra, "SOA testing perspective model for regression testing". In Proceedings of 2013 Nirma University International Conference on Engineering (NUiCONE).

Authors' Profiles

Abhishek Kumar passed his M.Tech from school of computer engineering, KIIT University, Bhubaneswar, Orissa, India. He joined Trinity institute of technology & research, Bhopal (M.P) as an assistant professor in computer science department. His area of interest is software testing, web services and software designing.

Manindra Singh passed his B.Tech in CSE. He is RHCE certified Engineer and also Associate Member of Institute of Engineers(Kolkata). His research area include linux system, Apacha server and Hardoop.

How to cite this paper: Abhishek Kumar, Manindra Singh,"An Empirical Study on Testing of SOA based Services", International Journal of Information Technology and Computer Science(IJITCS), vol.7, no.1, pp.54-66, 2015. DOI: 10.5815/ijitcs.2015.01.07