Black Box Test Case Prioritization Techniques for Semantic Based Composite Web Services Using OWL-S

A.Askarunisa[#1], K.Arockia Jackulin Punitha[*2], A.M.Abirami[#3]

[#] *Department of Computer Science, Thiagarajar College of Engineering*
Madurai, India
[1]aacse@tce.edu
[3]abiramiam@tce.edu

[*]*Thiagarajar College of Engineering*
Madurai, Tamilnadu, India
[2]punitha_charlas@tce.edu

***Abstract*— Web services are the basic building blocks for the business which is different from web applications. Testing of web services is difficult and increases the cost due to the unavailability of source code. Researchers have, web services are tested based on the syntactic structure using Web Service Description Language (WSDL) for atomic web services. This paper proposes an automated testing framework for composite web services based on semantics where the domain knowledge of the web services is described using protégé tool [4] and the behaviour of the entire business operation flow for the composite web service is described by Ontology Web Language for services (OWL-S)[1]. Prioritization of test cases is performed based on various coverage criteria for composite web services. Series of experiments were conducted to assess the effectiveness of prioritization and empirical results shown that prioritization techniques perform well in detecting faults compared to traditional techniques.**

***Keywords*— Average percent of Faults Detected, Composite web services, Harmonic Mean of TF, Harmonic Mean of Sequence Invocation, Ontology Web Language for services, Protégé, Test Case Prioritization.**

I. INTRODUCTION

Web services [6, 8] are an enabling technique for Service Oriented Computing which provides W3C standard based mechanism and open platform for integrating distributed autonomous service components. The quality of services is a key issue for developing service-based software systems and testing is necessary for evaluating the functional correctness, performance and reliability of individual as well as composite services. WSDL [12] an XML file describes the web service through which the location and operations of the web service requested by the client are identified from Universal Description Discovery and Integration register (UDDI). Unfortunately, WSDL descriptions only addresses the functional aspects of a web service without containing any useful description of non-functional or Quality of Services (QOS) characteristics.

The semantic web [5, 1, 7] is an evolving development of the World Wide Web (WWW) in which the meaning (semantics) of information and services provided on the web is defined, making it possible for the web to understand and satisfy the requests of people and machines to use the web content. WSDL-S and OWL-S aim to support the use of semantic concepts in Web service discovery [2], interoperability and composition. Ontology is used to make the Web Services understandable for computers and thus support automatic discovery, invocation and composition of Web Services. WSDL-S [5] relies on external ontology semantics but cannot provide pre and post constraints for operations present in the WSDL.

OWL-S [1] includes its own semantics through OWL where Split-Join, Repeat-While, Split-Perform and sequence can be provided for operations in WSDL. In this paper, OWL-S is used to define the sequencing of web service operations for composite web services, test cases generated based on sequences, coverage computed for the test cases and prioritization of test cases is performed to improve the effectiveness of regression testing.

II. RELATED WORK

In literature, there is a significant amount of research activities that had been done in web services testing to maintain the quality. Chunyan Ma et al. [12] Dealt with the test data generation for atomic web service. Testers can automatically generate test data to satisfy the requirement in a web service. Yongbo Wang et al. [2] discussed about the test case generation of semantic web services based on ontology. It was done by traversing the Petri-net to produce test steps and test data. Test data was generated based on ontology over Input Output Preconditions and Effect (IOPE) analysis.

Siripol Noikajana and Taratip Suwannasart [5] discussed about WSDL-S with combinatorial testing. This method of testing is the selection of test case based on the input combination. Every combination of input is covered at least by one test case. OCL is a formal language which was used to describe an expression on UML models. These expressions specify the invariant conditions for atomic web services. II Woong Kim and Kyong-Ho Lee [1] discussed about OWL-S

978-1-4577-0590-8/11/$26.00 ©2011 IEEE

equivalent terms for the UML diagrams. Unified Modelling Language (UML) model is used to represent the business activities. OWL-S was created in the XML Style sheet Language Transformation (XSLT) format and this specification was used for representing the OWL-S composition manually in this work. Xiaoying Bai and Shufang Lee [7] discussed about ontology based testing of web services. The issues in web service testing are contract based collaborative testing and automated testing and test case generation. To overcome these issues test ontology model described by OWL-S is used.

Mei et al. [11, 10] used the mathematical definitions of XPath as rewriting rules, and proposed a data structure known as an XPath Rewriting Graph (XRG). They have also studied the problem of black-box test case prioritization of services based on the coverage information of WSDL tags [10]. From the above literature, there is less/no work on testing composite web services. This paper proposes partially automated model for testing composite web services. The model includes semantics through OWL-S to generate sequences of the composite web services.

III. BACKGROUND

The following section details on the background materials required for including semantics with the help of OWL-S.

A. OWL-S Description

The OWL-S contains three parts [1]. They are

1) Service Profile: Describes the various functions provided by the web service to the user and who provides the service The profile ontology defines the functionality description through various properties like hasParameter - states the domain knowledge explicit, hasInput - input range as defined in the process ontology, hasOutput - output range as defined in the process ontology, hasPrecondition-specifies the preconditions defined in the process Ontology, hasResult - range of results as defined in the process ontology.

2) Service Grounding: Describes how to interact with the service with details of transport protocols and access details of port number and to link the service with its elements of Service Profile and Service Model. Service Grounding uses the service class that has the following elements like Presents – provide link to the Service Profile, Described By – provide link to the Service Model, Supports – provide link to Service Grounding.

3) *Service Model:* Describes how the service can be used, how it works, what are preconditions and post conditions and input and output of the service. There are three process levels. Atomic processes are evocable process that is evocated by giving the input, process the input and give the output. They generally correspond to WSDL operations. Simple processes are abstracted process. Composite process is combination of simple and atomic processes give the composite process which can be further classified into simple and complex composite web service based on its composition. The complexity of composite process is explained below. Service requestor requests a composite web service by passing SOAP messages. The composite web services are not available in UDDI. Service provider provides composite web services by adding semantics to the atomic web services. Dataflow and control flow (blue) are in atomic web services except semantic flow (orange) in composite web services.

Fig. 1 Model for Composite Process

In Fig. 1, M1, M2…..Mn are parameters and A1, A2, A3……An are operations which is shown in WSDL for each web service WS1, WS2……..WSn. In this paper, we have developed a complex composite web service called Weather Monitoring System (WMS) by including logical sequence flow between web services (semantics) through OWL-S.

B. Weather Monitoring System(WMS)

Weather Monitoring Service (WMS) [9] is a complex composite web service that includes the following web services, ValidateEmail, PhoneVerify, ValidateCreditcard, GetCityByCountry, GetWeatherByCity which is used in the experiment.

The WMS initially checks for the validity of the email-id. If given email is invalid then it verifying the mobile number. If the mobile number is valid then credit card details are verified, then check whether the city to be visited is a valid city of a particular country. If all the above sequences validate to be true, the WMS gives information about the weather conditions of the requested city. The flow of the logical sequences among the various services of WMS is as shown Fig 2. A node in Fig. 2 represents an activity and a link represents a work flow transition between two activities. The activities are labelled as Ai for i = 1 to 11.

The weather monitoring service needs to invoke all the web services to handle the user's trip arrangement request. A test

on such a service is usually done by sending request messages followed by receiving and handling response messages.

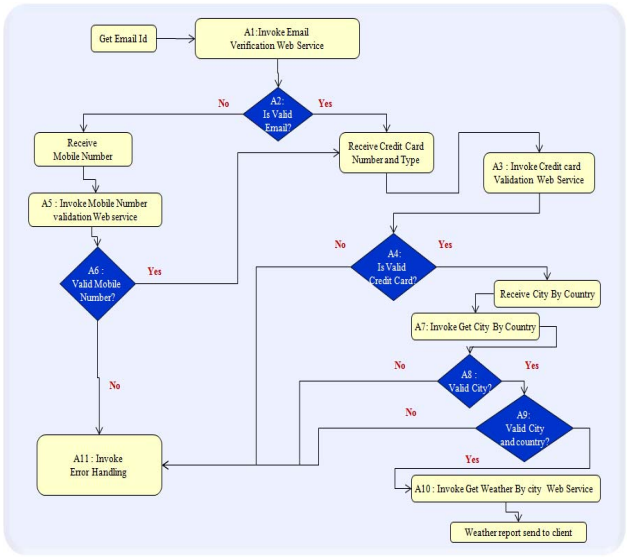

Fig. 2 Work flow for WMS

IV. IMPLEMENTATION

The complete process of including semantics, test case generation and prioritize is implemented using Net beans based on the model given in Fig. 3.

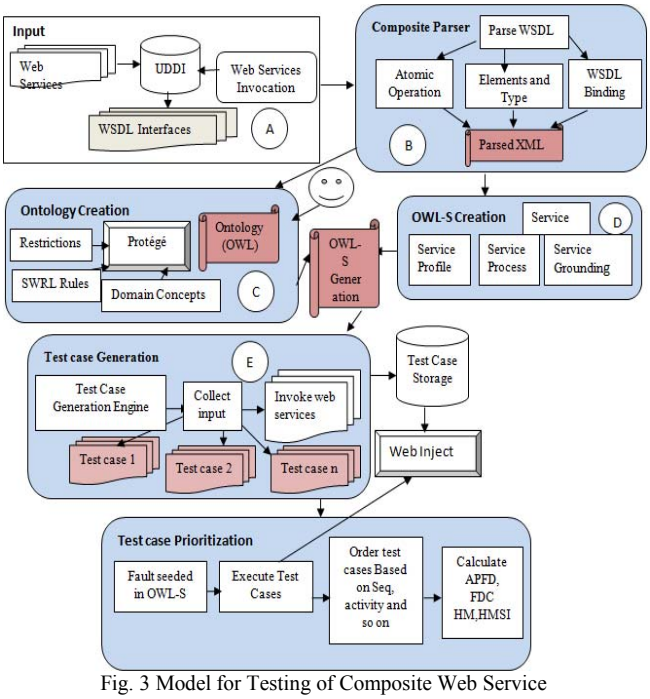

Fig. 3 Model for Testing of Composite Web Service

The web service to be tested is parsed so able to get the information needed for invocation. The domain ontology includes the static description of the parameters, constraints to the web service and the activity flow for composite web services provided by OWL-S. Test cases are automatically generated and executed to give the result required to the user. Test cases are prioritized for effectiveness. The detailed functional description of each module is explained below.

A. *Extraction of WSDL*

The required web services which are to be made composite are identified from UDDI registry. The web services publicly available in the form of WSDL interface are extracted. The complex composite web service includes many atomic web services and composite web service. This model is testing complex composite web services also.

```
<?xml version="1.0" encoding="UTF-8"
standalone="no" ?>
- <opList>
- <servicename name="ValidateEmail">
  <URL
name="http://www.webservicex.net/ValidateEmail.asmx"
/>
  <operationname name="IsValidEmail" />
  <parameters name="Email" type="s:string" />
  </servicename>
- <servicename name="ValidateEmail">
  ----------
```

Fig. 4 Output of composite WSDL parser

Since the WSDL generated contains information recursively and is quite lengthy, it is parsed using Java XML Processing (JAXP) APIs to retrieve the Parameters, its data types for each atomic process and its WSDL binding information useful for test case generation. For example, ValidateEmail web service is parsed and output shown in Fig. 4. In this, the input parameter is Email for the atomic process IsValidEmail and its WSDL binding is **http://www.webservicex.net/IsValidEmail** and similarly shown for the output operation. This information is parsed for all selected web services and stored in the XML file format as shown in Fig. 4.

B. *OWL-S Generation*

The semantics are included in two ways such as domain ontology and process ontology. In domain ontology, the static hierarchical relationships among web service concepts and their constituent properties are represented with a Class diagram and the restrictions for their properties are provided by the SWRL rules. Protégé tool [4] is used to create the ontology and verifies syntactic correctness as well as checks the semantic constraints specified. This model is used to reuse existing OWL ontology.

The process ontology describes the behaviour of operational flow between the web services that is the flow of control from one activity to other activity. In this, activity represents the services to be invoked and their validation of each atomic process. The behaviour of the entire business activity is shown by the work flow in the form of activity diagram of the web services, as shown in Fig, 2. The ontology created in the previous module is referred and included in the

OWL-S output. OWL-S is generated for WMS as shown in Fig 5.

```
<owl:Ontology rdf:about="">
    <owl:imports
rdf:resource="http://sqwrl.stanford.edu/ontologies/built-
ins/3.4/sqwrl.owl"/>
    <owl:imports
rdf:resource="http://swrl.stanford.edu/ontologies/3.3/swrl
a.owl"/>  --------
        <owl:Class>
      <owl:unionOf rdf:parseType="Collection">
        <owl:Class
rdf:about="#isValidresponse"/>
        <owl:Class
rdf:about="#validateCardNumberResponse"/>
    --------
<owl:Class rdf:ID="GetWeatherResponse">
    <rdfs:subClassOf rdf:resource="#GetWeather"/>
<rdf:resource="#GetWeatherResult"/>
<owl:allValuesFrom
rdf:resource="#GetWeatherResponse"/>          -----
rdf:ID="GetWeatherResult">                    -------
rdf:resource="#GetWeatherResponse"/>          -------
<rdfs:range>rdf:resource="#ValidateCardNumber"/>-----
----------------------
```

Fig. 5 OWL output

C. Test Case Generation

The order of web services to be invoked by checking the pre conditions present in OWL-S are collected. The test case generator GUI is developed using Net beans and checked in online. The connection of web service is made at run time by corresponding URL. The URL for invoking web service is automatically generated from the parsed information shown in Fig. 4. For Example, ValidateEmail web service has one input parameter as email and input operation is ValidateEmail. The URL = http:// WWW. Web servicex.net/ValidateEmail.asmx? Email=”input” [9] is created at runtime and invoke the web service. This is stored as the test case for the corresponding composite web service and sample test cases for WMS are shown in Fig. 6. The result of the web service is stored in the XML file and verified as valid or not. If it is valid then continue the invocation of web services described in OWL-S.

```
<Testcases>
<case id="0" description1="Case should Success"
method="get"
url=http://www.webservicex.net/ValidateEmail.asmx/IsVa
lidEmail?Email=skraro@yahoo.co.in          Verify
Positive="valid"/>
<case id="1" description1="Case should Success"
method="get"
url=http://www.webservicemart.com/phone3t.asmx/Pho
neVerify?PhoneNumber=77011946923           Verify
Positive="true"/>
```

Fig. 6 Output of Test Case Generation

The test cases are partitioned based on the number of successful invocation of web services. Then each partition is stored in separate XML file. After the partition, any one test case from the partitioned XML file is selected to test a particular sequence of invocation which reduces the number of repeated test cases and the time of testing. Web Inject tool is used to test web service based on HTTP request.

D. Test Case Prioritization

Test case prioritization [10, 13] techniques schedule test cases in an execution order according to some criterion. The purpose of this prioritization is to increase the likelihood that if the test cases are used for regression testing in the given order, they will more closely meet some objective than they would if they were executed in some other order. Test case prioritization can address a wide variety of objectives, such as testers may wish to increase the rate of fault detection and to increase their confidence in the reliability of the system under test at a faster rate. The problem statement for prioritization is [13],

Given: T is a test suite; PT is the set of permutations of T; f is a function from PT to the real numbers.

Problem: Find T' Є PT such that (T" Є PT) (T" ≠ T’) [f (T') ≥ f (T")]

Here, PT represents the set of all possible prioritizations (orderings) of T and f is a function that, applied to any such ordering, yields an award value for that ordering.

This paper performs test case prioritization based on the number of activities present, number of tags /elements present, number of transitions present [11], number of sequences accessed, fault rate and fault severity which is explained below. Many WSDL documents define XML schemas used by services. Each XML schema contains a set of elements. Intuitively, the coverage information on these WSDL tags reveals the usage of the internal messages among activities. The paper considered eight different prioritization techniques classified into five groups. Table I lists the various techniques by groups.

TABLE I
TEST CASE PRIORITIZATION TECHNIQUES

Technique	Group Name	Description
WST1 WST2	Benchmark	No Prioritization Random
WST3 WST4	Activity Based	Tot-Activity Tot-Transition
WST5	Tag based	WSDL-Tags
WST6	Sequence based	Tot-Seq
WST7 WST8	Fault Rate Based	Fault Rate Severity Based[15]

In most of the prioritization study, random and No prioritization techniques are considered mainly for comparison. This is an experimental control technique. When a test case is executed for a web service, the result of

execution will involve a number of activities and transitions [14]. Test cases are ordered based on the number of activities covered (WST3) and transitions covered (WST4) which is similar to the statement coverage and branch coverage respectively.

Prioritization on activity coverage selects a test case with maximum number of activities covered and the next maximum. Ordering of test cases are based on their activity coverage affects the rate of fault detection of the ordered test suite. As the number of requests in a test case partially determines how much of the tags [10](WST5) and sequence (WST6) are exercised by the test case, ordering of test cases based on their tag and sequence coverage affects the rate of fault detection of the ordered test suite.

By WST7 technique, test cases are prioritized based on the fault rate, where fault rate is determined by the ratio of the total number of faults detected by the test case t_i to the execution time taken by t_i. In WST8 technique [21], the weight of the test case is calculated based on fault rate and the fault impact of a test case. Test cases are sorted for execution based on the descending order of test case weight, such that test case with highest test case weight runs first.

V. Result and Analysis

The effectiveness of the prioritization strategies were evaluated by their rate of fault detection. Experiments were conducted in the study to meet the following objectives.

A. To check whether the proposed test case prioritization techniques improve the rate of fault detection capabilities for composite web services test cases.

B. To select the best prioritization technique based on metrics.

For each approach controlled experiments were conducted and the study considered different web services which are frequently used. The study is considered two applications with faulty versions. The faults are inserted [3] into the OWL-S due to unavailability of source code of web services. The subject programs have different characteristics. The compositions between web services was created and tested. The details of these applications are as follows:

WMS: This web service includes many web services ValidateEmail, PhoneVerify, CreditCardChecker, GetCitiesByCountry, GetWeatherByCity [9] and the like. In the WMS, the above said web services may either work atomically or may be combined to work in composition.

Bible [9]: This web service contains many other web services GetWordsBy Chapterandverses GetWords ByBook Titleand Versus and the like.

Conditions were included and sequences were generated to make all the services composite by the provider. Test cases were generated and prioritized by different techniques. The details of different application are shown in TABLE II.

TABLE III
SUBJECT APPLICATIONS AND THEIR CHARACTERISTICS

S.No	Metrics	WMS	Bible
1.	Number of Test cases	25	30
2.	Number of activities	15	6
3.	Number of transitions	13	3
4.	Number of tags	19	4
5.	Number of sequences	5	3
6.	Number of faults	12	6

To perform and evaluate the various prioritization techniques, the following are required.

1) Web Services to be tested.
2) Test cases for the services.
3) Faults associated with the service in the form of a fault matrix.
4) Execution time associated with every test case.

The test cases were generated based on customer requirements. For the fault seeding process, the seeded faults like changing enumerators, changing logical operation and the like. Each of the prioritization technique was implemented in Java and executed twenty five times and the average of the values considered for analysis. The complete model (each component) was implemented using Net Beans.

A. Checking Fault Detection Capability

For meeting the first objective, to check whether the proposed test case prioritization techniques improve the rate of fault detection capabilities for composite web services test cases, the following performance metrics are evaluated

1) Average percent of Faults Detected (APFD)
2) Harmonic Mean of TF (HM_{TF})
3) Harmonic Mean of Service Invocations (HM_{SI})

1) Average Percent of Faults Detected:

Rate of Fault Detection of Prioritized Test Suite to quantify the goal of increasing a test suite's rate of fault detection, the metric, Average Percentage of fault detected (APFD), which measures the weighted average of the percentage of faults detected. APFD values generally ranges from 0 to 100; higher numbers imply faster (better) fault detection rates. The APFD for test suite T0 is given by the equation,

$$APFD = 1 - \left(\frac{TF1+TF2+\cdots+TFm}{nm}\right) + \left(\frac{1}{2n}\right) \quad (1)$$

Where TFi is the position of test case detecting fault 'i', 'm' is the number of faults and 'n' is the number of test cases.

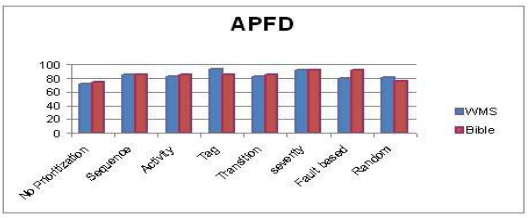

Fig. 7 Comparison of APFD Values

The reduced test cases are prioritized based on no prioritization, random, sequence coverage, activity coverage, transition coverage, tag coverage, severity based coverage and fault rate and the corresponding APFD is calculated. The following graph shows the APFD values for different application for different criteria. From the Fig. 7, the average

APFD value for TCP based on severity based rate is 91.67% than WST1 which is 73.35%.

2) Harmonic Mean for TF:

Harmonic Mean (HM_{tf}), which is independent of the test suite size. HM is a standard mathematical average that combines different rates into one value. The Harmonic Mean value is calculated by using the formula as shown below,

$$HMtf = \frac{tn}{\frac{1}{TF1} + \frac{1}{TF2} + \cdots + \frac{1}{TFm}} \quad (2)$$

Where T is a test suite consisting of n test cases and F be a set of m faults revealed by T. TFi be the first test case in the reordered test suite S of T that reveals fault i. Fig. 8 shows the different HM_{tf} values for various applications for proposed techniques. For all the applications, the severity based technique (WST8) shows improvement and higher HM_{tf} value than other techniques. The Severity based technique has 11.64 as average.

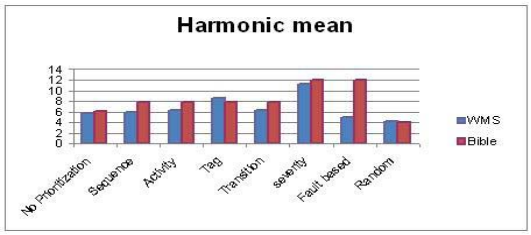

Fig.8 Comparison of Harmonic Mean for Web services

3) Harmonic Mean Sequence Invocation:

Harmonic Mean Sequence Invocation is used for web services specifically which is independent of the test suite size. It is based on the number of web service invoked. The formula for this is shown below,

$$HMsi = \frac{tn}{\frac{1}{SI1} + \frac{1}{SI2} + \cdots + \frac{1}{SIm}} \quad (3)$$

Where T is a test suite consisting of n test cases and F be a set of m faults revealed by T. SIi be the number of invocation of web services in the reordered test suite S of T that reveals fault i. Fig. 9 shows HMsi values for all applications for proposed techniques. From the figure, it is clear that the severity based technique has the maximum value of 30 compared to other techniques,

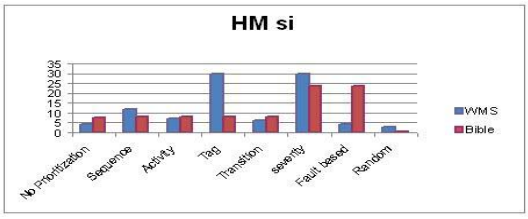

Fig. 9 Comparison of Harmonic Mean Sequence Invocation for Web services

B. Best Prioritization Technique

From the various metric calculations, the severity based prioritization technique of composite web services is better than other benchmark techniques in fault deduction.

VI. Conclusion

In this paper we have proposed automated model for testing of composite web services, where the logical sequence flow between the web services (semantics) is included through OWL-S. We have proposed various techniques for prioritization of test cases, generated for the composite web services. Experiments are conducted and the evaluation of techniques is performed based on APFD, HM_{tf} and HMsi. The results show that the fault severity based technique has a higher rate of fault detection, higher harmonic mean based on test case fault and service invocation.

In the future, the prioritization will be done by using other indexes such as the resource, Cost based and the like. The Composition can be extended with BPEL (Business Process Execution Language) for improving as per the industry standard.

References

[1] Il Woong Kim and Kyong - Ho - Lee, "A Model - Driven Approach for Describing Semantic Web Services: From UML to OWL-S", IEEE transactions on System, man and cybernetics, - Part C: Applications and reviews, vol 39, No 6, November 2009.

[2] YongboWang, Xiaoying bai, Juanzi Li, and Ruobo Huang, "Ontology based test case generation for Testing Web Services", 8th International Symposium on Autonomous decentralized Systems, 2007.

[3] Xiaojuan Wang, Ning Huang and Rui Wang, "Mutation Test Based on OWL-S Requirement Model", IEEE International Conference on Web Services, 2009.

[4] protégé Tool: http://protege. stanford.edu/ download.html

[5] Siripol Noikajana and Taratip Suwannasart, "An Improved Test Cases Generation Method for Web Services Testing from WSDL-S and OCL with Pair-Wise Testing Technique", 33 rd Annual IEEE international computer Software and Applications Conference, 2009.

[6] Massimo paolucci and MatthiasWanger, "Grounding OWL-S in WSDLS", IEEE International Conference on Web Services, 2006.

[7] Xiaoying bai and Shufang Lee, Wei Tek Tsai and Yinong Chen, "Ontology Based Test Modelling and Partition Testing of Web Services", IEEE International Conference on Web Services, 2008.

[8] Sapna Malik, Sanjay Kumar Malik, Nupur Prakash, SAM Rizvi,"Comparative Study of Technologies of Semantic Web Services: OWL-S, WSMO and WSDL-S".

[9] Web Service: http:// www. webservicex. net/ ValidateEmail. asmx?wsdl.

[10] Lujun Mei, W.K.Chan, T.H. Tse, Robert G. Merkel, "Tag – Based Techniques for Black-Box Test Case Prioritization for Service Testing", 9th International Conference on Quality Software, 2009.

[11] Lujun Mei, W.K.Chan, T.H. Tse, Zhenyu Zhang," Test Case Prioritization for Regression Testing of service - Oriented Business Applications", International conference on web Engineering,2009.

[12] Chunyan Ma, Chenglie Du, Tao Zhang, Fei Hu, Xiaobin Cai, "WSDL Based Automated Test Data Generation for Web Service", International Conference on Computer Science and Software Engineering, 2008.

[13] G. Rothermel, R. Untch, C. Chu, and M. Harrold, "Test CasePrioritization," IEEE Transactions on Software Engineering, vol. 27, pp. 929-948,October 2001.

[14] Ke Zhai, Bo Jiang, W.K.Chan, T.H.Tse, "Taking Advantage of Service Section: A Study on the Testing of Location Based Services through Test Case Prioritization", ICWS 2010.

[15] R.Kavitha and N.Sureshkumar, "Test Case Prioritization for Regression Testing based on Severity of Fault ", (IJCSE) International ournal on Computer Science and Engineering Vol. 02, 1462-1466, No. 05, 2010.