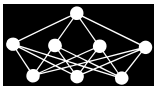

TEST CASE GENERATION AND PRIORITIZATION FOR COMPOSITE WEB SERVICE BASED ON OWL-S

A. Askarunisa[*], *K. Arockia Jackulin Punitha*[†], *N. Ramaraj*[‡]

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 coder. In previous work, web services were 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 semantics, where the domain knowledge of the web services is described by protégé tool [13] and the behavior of the entire business operation flow for the composite web service is provided by Ontology Web Language for services (OWL-S)[6]. Prioritization of test cases is performed based on various coverage criteria for composite web services. Series of experiments were conducted to assess the effects of prioritization on the coverage values and benefits of prioritization techniques were found.

Key words: *Composite web services, ontology web language for services, protégé, test case prioritization, web services*

Received: December 20, 2010
Revised and accepted: November 23, 2011

1. Introduction

Web services [10, 15] 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.

[*]A. Askarunisa
Thiagarajar College of Engineering, Department of Computer Science, Madurai, India, E-mail: aacse@tce.edu
[†]K. Arockia Jackulin Punitha
Computer Science Department in Thiagarajar College of Engineering, Madurai, India, E-mail: punitha_charlas@tce.edu
[‡]N. Ramaraj
GKM College of Engineering and Technology, Chennai, India

WSDL [2] is an XML file describing the location and operations of the web service requested by the client which are identified from the Universal Description Discovery and Integration register (UDDI). Unfortunately, WSDL descriptions only address the functional aspects of a web service without containing any useful description of non-functional or Quality of Services (QOS) characteristics.

The semantic web [4, 10, 16] is an evolving development of the World Wide Web (WWW) in which the meaning (semantics) of information and services provided on the web are 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 [21], 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 [16] relies on external ontology semantics but cannot provide pre and post constraints for operations present in the WSDL. Pre and post constraints are used to indicate the compositions of web service and the logical flow between web services.

The web service semantics (WSDL-S) aims to add semantic annotation to web service description by extending WSDL. WSDL-S is an extension of the syntactical level of WSDL and includes semantic capabilities for semantic web services [16]. WSDL-S associates the semantic descriptions to the web service in order to enable automatic search, discovery, selection, composition and integration across heterogeneous users and domains. WSDL-S includes three attributes and two elements, in addition to that of WSDL. They are:

- The **precondition** element is a set of assertions that must be met before web services can be invoked.
- The **effect** element is an element that is a result of invoking a web service operation.
- The **modelReference** attribute is a specification of association between WSDL entity and a concept.
- The **schemaMapping** attribute is a handling structure which differentiates between schema elements of web services and their corresponding semantic model concepts.
- The **category** attribute is a service categorization of information for publishing a service in a web services registry.

OWL-S [6] includes its own semantics through Ontology Web Language (OWL) where Split-Join, Repeat While, Split Perform and sequence can be provided for operations in WSDL. There are three parts in OWL-S. They are: service profile which is the high level description of the web service. Service model provides service capabilities and Service grounding is the mapping from the abstract to concrete specification. OWL-S provides the details about how to invoke web service from the composite process. Ontology (OWL) is used to make the web services understandable for computers and thus support automatic discovery, invocation

and composition of web services. The semantics provide the composition description of the web service invocation. It is provided by specifying the constraints in invocation in OWL-S.

This paper uses OWL-S 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 testing.

2. Related Work

In literature, a significant amount of research activities had been done in web services testing. Chunyan Ma et al. [2] 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. [21] discussed the test case generation of semantic web services based on ontology. It is done by traverse Petri-net to produce test steps and test data. It is generated based on ontology over Input Output Preconditions and Effect (IOPE) analysis.

Siripol Noikajana and Taratip Suwannasart [16] discussed WSDL-S with combinatorial testing. This method of testing is the selection of test case in the input combination. Every combination of input is covered by at least one test case. It is a formal language used to describe expressions on UML models. These expressions typically specify invariant conditions that must hold for the system being modeled or queries over objects described in a model. OCL can be used for a number of different purposes:

- As a query language
- To specify invariants on classes and types in the class model
- To describe pre and post conditions on operations and methods

II Woong Kim and Kyong-Ho Lee [6] discussed the OWL-S equivalent terms for the UML diagrams. Unified Modeling Language (UML) model is used to represent the business activities. OWL-S is created in the XML Style sheet Language Transformation (XSLT) format and this specification is used for representing the OWL-S composition manually in this work.

Xiaoying Bai and Shufang Lee [20] discussed 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 the first issues, test ontology model is used, and to overcome the second issue, ontology is created by the OWL-S using semantics.

Mei et al. [8, 9] used the mathematical definitions of XPath as rewriting rules, and proposed a data structure known as an XPath Rewriting Graph (XRG). They have also proposed a hierarchy of prioritization techniques for regression testing of services by considering different levels of business process, XPath, and WSDL information as the methods to resolve conflicts faced by prioritization. They have also studied the problem of black-box test case prioritization of services based on

the coverage information of WSDL tags [8]. They have explored the XML message structure exchanged between services to guide test case prioritization.

In our previous work, we have generated test cases for Web Services using reduction techniques Pair-Wise Testing (PWT) and Orthogonal Array Testing (OAT) and compared the two techniques. Orthogonal Array Testing is a systematic and statistical way of testing. Orthogonal arrays could be applied in user interface testing, system testing, regression testing, configuration testing and performance testing. Orthogonal arrays are the extension of Latin Squares. A pair-wise testing (PWT) technique is one of the combinatorial testing techniques which considers uniqueness on a pair of input parameters. Every combination of their valid values should be covered by at least one test case in the test suite.

The structures of Web Services were specified using UML diagrams. The pre and post conditions for the service rule were specified using Object Constraint Language (OCL). The framework developed transformed this into WSDL-S specifications which were parsed and transformed into structured DOM tree. The Document Object Model (DOM) is an application programming interface for well-formed XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. Test data set generated by the framework satisfied the constraints of the WSDL and test cases were developed based on the data generated, documented in XML based test files called Web Service Test Specifications (WSTS) and executed. The number of test cases required by general testing, PWT, OAT were compared and the better testing technique for testing Web Services is determined. In this paper, we have enhanced the inclusion of semantics through OWL-S, for a composite web service, which provides the flexibility in allowing the creation of many grounding rules.

From the above literature, there is less/no work on testing composite web services. Thus, 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.

3. Background

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

3.1 OWL-S description

The OWL-S contains three parts [6]. 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* – which 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, *hasResu*lt – 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. To link the service with its elements of Service Profile, Service Model and 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 – Atomic processes are evocable processes that are evocated by giving the input, process the input and give the output. They generally are corresponding to WSDL operations.

Simple processes – abstracted processes.

Composite process – combination of simple and atomic processes gives 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 detailed 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 are in atomic web services except for semantic flow in composite web services.

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.

3.2 Weather Monitoring System (WMS)

Weather Monitoring Service (WMS) [18] is a complex composite web service that includes the following web services, *ValidateEmail, PhoneVerify, ValidateCreditcard, GetCityByCountry, GetWeatherByCity.*

The WMS initially checks for the validity of the email-id. If a given email is invalid then it verifies the mobile number. If the mobile number is valid then credit card details are verified, then checked 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 labeled 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.

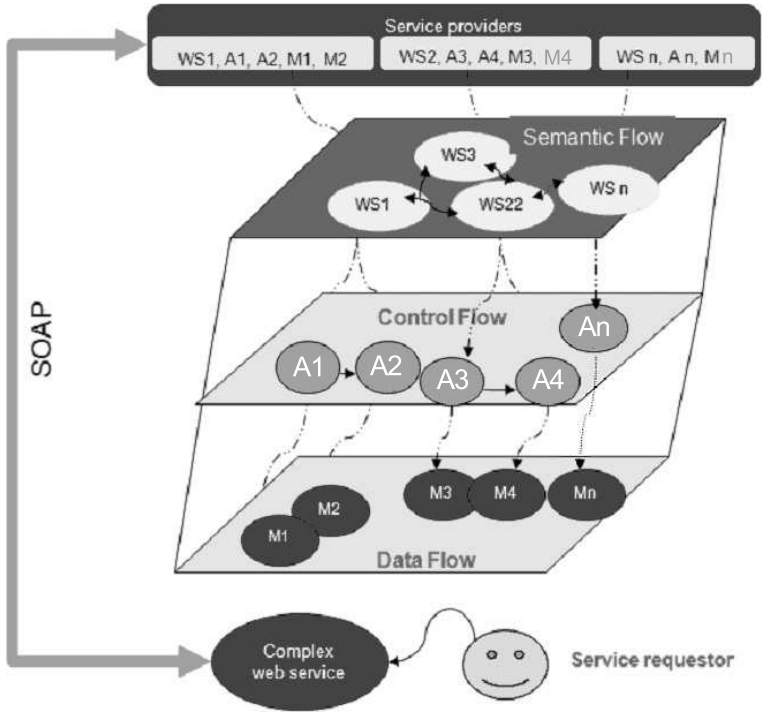

Fig. 1 *Model for composite process.*

4. Test Case Prioritization

Test case prioritization [3, 8] 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 [3].

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 $T' \in PT$ such that $(T'' \in PT)\ (T'' \neq T')\ [f\ (T') \geq f\ (T'')]$

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 [9], number of sequences accessed, fault rate and fault severity which are detail below.

Many WSDL documents define XML schemas used by services. Each XML schema contains a set of elements. Intuitively, the coverage information on these

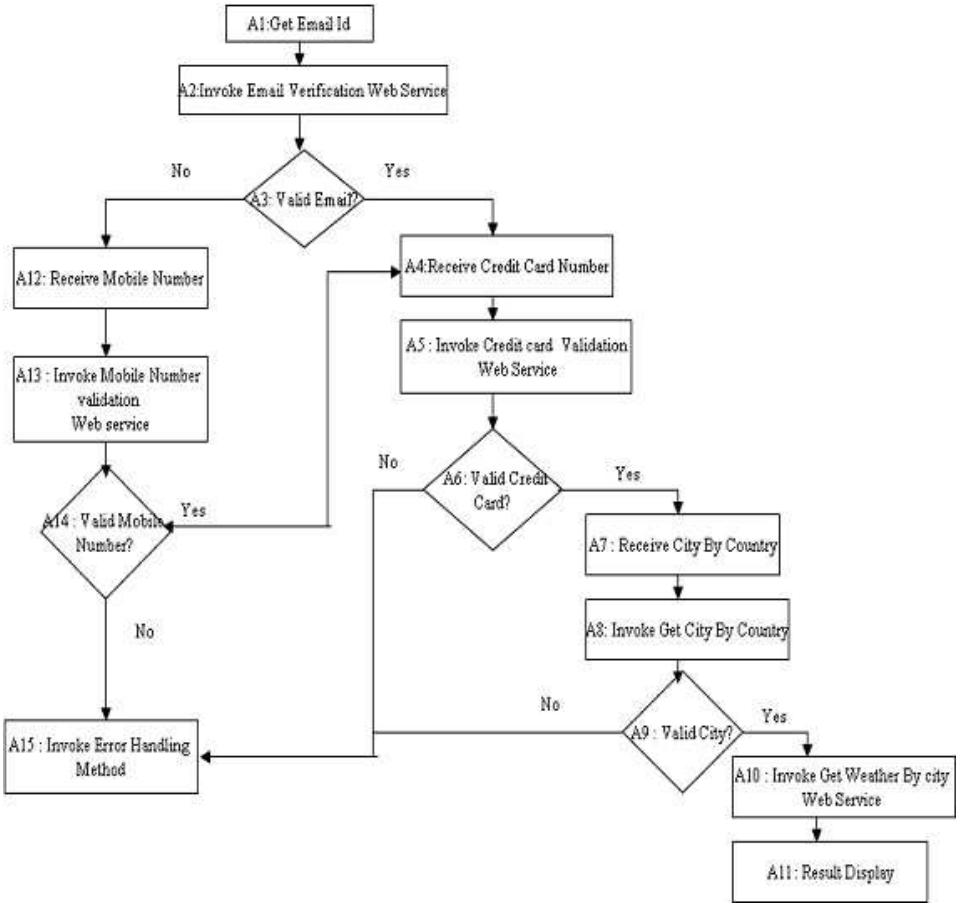

Fig. 2 *Work flows of WMS.*

WSDL tags reveals the usage of the internal messages among activities. The paper considered eight different techniques classified into five groups. Tab. I lists the various techniques by groups.

a. Control Technique In most of the prioritization study, random (WST2) and No prioritization (WST1) techniques are considered mainly for comparison. This is an experimental control technique. In WST1 technique, test cases are taken without any prioritization. In WST2 technique, test cases are randomly selected for the prioritized test suite until there are no more test cases to select.

b. Activity based Techniques 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 based on their activity coverage affects the rate of fault detection of the ordered test suite.

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 Severitu Based[21]

Tab. I *Prioritization techniques.*

c. Element (Tag) based Techniques (WST5) Prioritization on tag coverage [8] selects a next test case with maximum number of tags covered. As the number of requests in a test case partially determines how much of the tags are exercised by the test case, ordering of test cases based on their tag coverage affects the rate of fault detection of the ordered test suite.

d. Sequence based Techniques As the number of requests in a test case partially determines how much of the web services are exercised by the test case, ordering of test cases based on their sequence count affects the rate of fault detection of the ordered test suite (WST6).

e. Fault-rate Prioritization 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 [14], 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.

5. Implementation

The complete process of including semantics, test case generation and prioritize then was implemented based on the model given in Fig. 3.

The web service to be tested is parsed which provide the information needed for invocation. The domain ontology includes the static description of the parameters and the constraints to the web service and the activity flow for composite web services provided by OWL-S. Test cases are generated and executed to give the

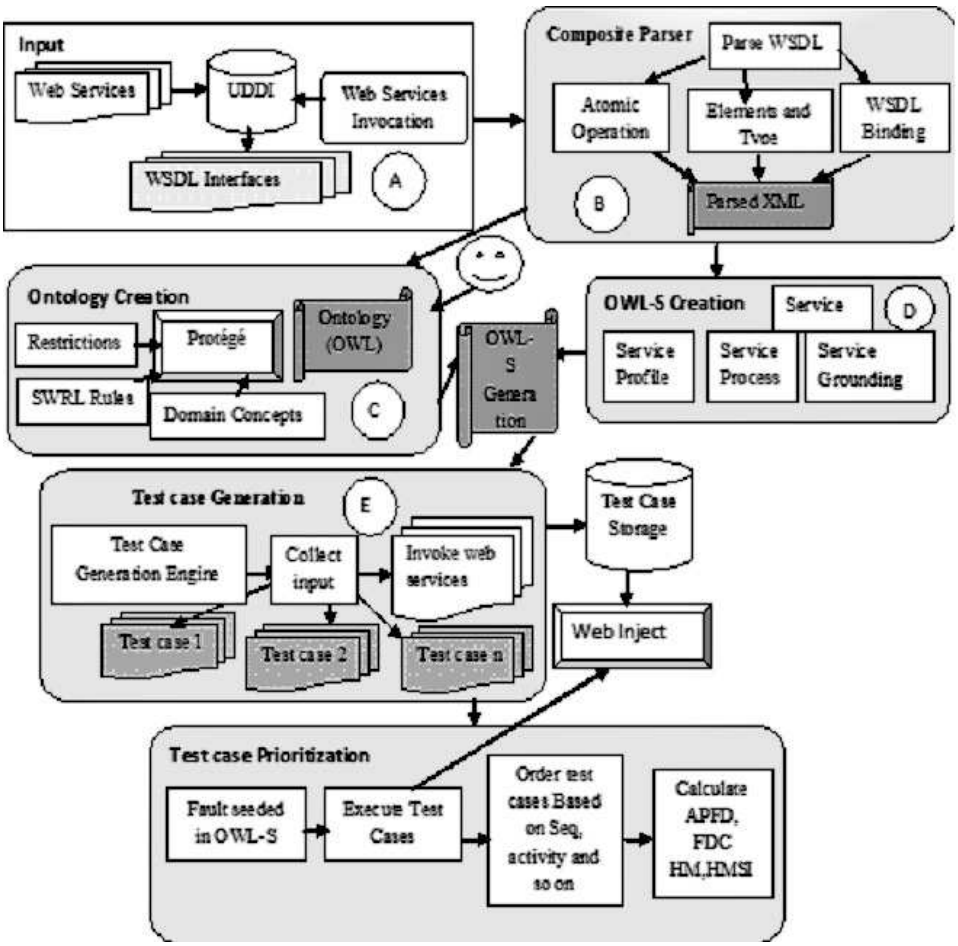

Fig. 3 *Model for testing of composite web service.*

result required to the user. Test cases are prioritized for effectiveness. The detailed functional description of each module is explained below.

5.1 Extraction of WSDL

The required web services to be made composite are identified from UDDI registry and their WSDL are extracted. The complex composite web service includes many atomic web services and composite web service.

5.2 Composite WSDL parser

The publicly available interface in the form of WSDL for the web service to be tested is obtained. Since the WSDL generated contains information recursively and is quite lengthy, it is parsed using Java XML Processing (JAXP) APIs to

retrieve the method signature useful for test case generation. The parameters and their corresponding data types for each operation are parsed and stored in the XML format as shown in Fig. 4.

```
<?xml version="1.0" encoding="UTF-8"
standalone="no"?>
<opList><servicename name="phone3T"><URL
name="http://www.webservicemart.com/phone3t.asm
x"/>
<operationname name="PhoneVerify"/>
<parameters name="PhoneNumber" type="s:string"/>
</servicename>
<servicename name="phone3T"><URL
name="http://www.webservicemart.com/phone3t.asm
x"/><operationname name="PhoneVerifyResponse"/>
<parameters name="PhoneVerifyResult"
```

Fig. 4 *Output of composite WSDL parser.*

5.3 Ontology creation

The semantics are included in two ways such as domain ontology and process ontology. In domain ontology, the hierarchical relationships among web service concepts and their constituent properties are represented with a Class diagram. The domain ontology describes the web service operation hierarchy in the composition and the restrictions for their properties are provided by the SWRL rules. The domain concept represents the static information about the operation and the data elements.

Protégé tool [5, 12 and 13] is used to create the ontology and will verify syntactic correctness as well as check that semantic constraints are specified. The existing domain ontology can be reused. The ontology output using protégé is shown in Fig. 5.

5.4 OWL-S Generation

The behavior of the entire business activity is shown by the work flow in the form of activity diagram of the web services.

The process ontology describes the behavior 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. The ontology created in the previous module is referred to and included in the OWL-S output. The OWL-S generator [11] developed with Net beans to generate OWL-S is shown in Fig. 6.

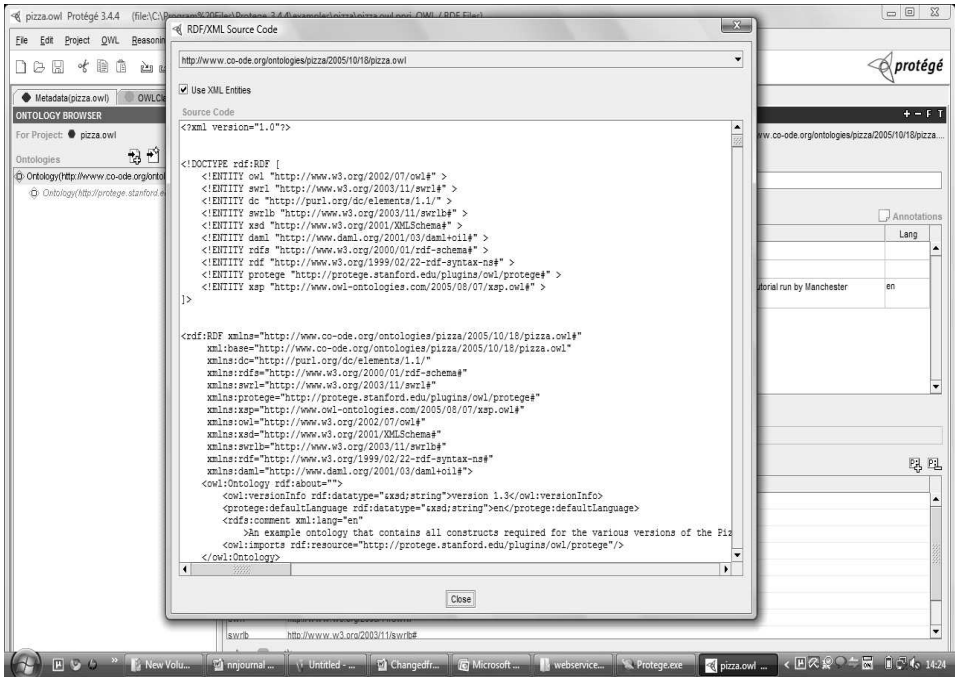

Fig. 5 *Output of Protégé tool.*

5.5 Test Case Generation

The order of web services to be invoked by checking the pre conditions present in OWL-S is collected. The Test case generator GUI is developed using Net beans. The connection of web service is invoked by corresponding URL and the result of the web service is stored in the XML file and verified as valid or not. The test cases are stored in the XML file after valid invocation.

The sample test cases are shown in Fig. 7. For example, the ValidateEmail web service has one input parameter as email. The URL = http://www.web servicex.net/ValidateEmail.asmx? Email="input" [18] is formed and connections made. 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 [17] is used to test web service based on HTTP request.

5.6 Test case prioritization

The test cases are prioritized based on the prioritization techniques detailed in Section 3.

```
<owl:Ontology rdf:about="">
      <owl:imports
rdf:resource="http://sqwrl.stanford.edu/ontologies/b
uilt-ins/3.4/sqwrl.owl"/>
      <owl:imports
rdf:resource="http://swrl.stanford.edu/ontologies/3.
3/swrla.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"/>
```

Fig. 6 *Output of OWL-S creation.*

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

Fig. 7 *Output of test case generation.*

6. Results and Discussions

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

- To check whether the proposed test case prioritization techniques improve the rate of fault detection capabilities for composite web services test cases.
- To select the best technique based on the average percent of faults detected ($APFD$).
- For each approach, controlled experiments were conducted and the study considered different web services which are frequently used.

6.1 Subject applications considered for analysis

The study has considered four applications with faulty versions. The faults are inserted [19] into the OWL-S due to unavailability of source code of web services. The subject programs have different characteristics. The compositions between web services were created and tested. The details of these applications are as follows:

WMS: This web service includes many web services: Validate Email, Phone Verify, Credit Card Checker, GetCities ByCountry, GetWeather ByCity [18] and the like. In the WMS, the above said web services may either work atomically or may be combined to work in composition.

Bible [18]: 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 Tab. II.

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

Tab. II *Subject applications and their characteristics.*

6.2 Experimental methodology

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 study has 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 was considered for analysis. The complete model (each component) was implemented using Net Beans.

6.3 Performance evaluation

The performances were evaluated based on the

- Average percent of Faults Detected ($APFD$)
- Harmonic Mean of TF (HM_{TF})
- Harmonic Mean of Service Invocations (HM_{SI})

6.4 Evaluation based on APFD

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 range from 0 to 100; higher numbers imply faster (better) fault detection rates. The $APFD$ for test suite $T0$ is given by the equation,

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

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.

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. Fig. 8 shows the area graphs drawn for proposed TCP techniques as stated in Tab. III.

Application	WST1	WST2	WST3	WST4	WST5	WST6	WST9	WST8
WMS	71.67	81.67	83.33	83.33	93.33	85	91.67	80
Bible	75	76.67	85	85	85	85	91.67	91.67

Tab. III *Average percentage of fault detection (apfd) for web services (in %).*

The following Tab. III shows the $APFD$ values for different application for different criteria. From the table, the average $APFD$ value for TCP based on severity based rate is 91.67% more than NoP which is 73.35%.

Fig. 9 shows the comparison of different techniques with different applications. It is clear from Fig. 9 that the severity (WST8) and tag (WST5) based methods detect faults at a faster rate than other techniques.

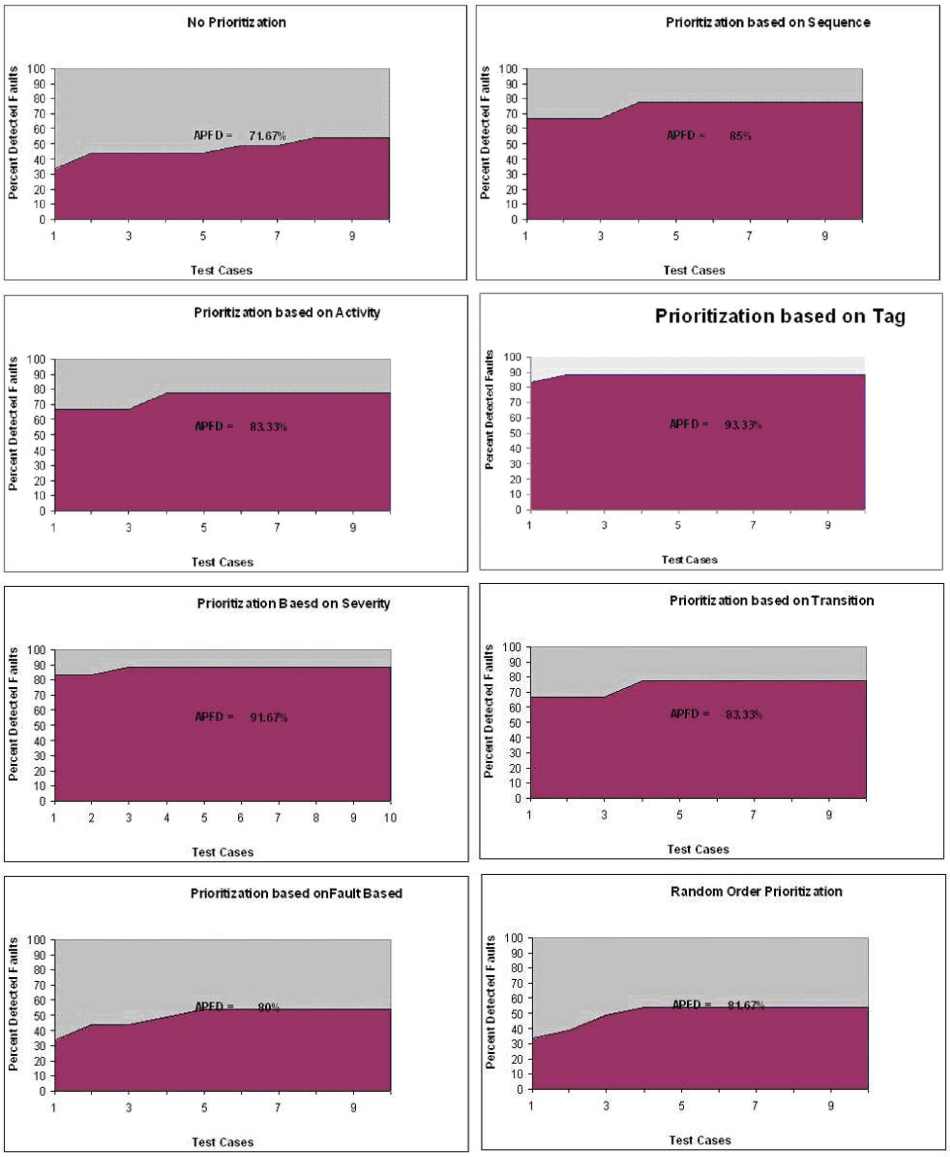

Fig. 8 *APFD Graph for various techniques.*

6.5 Harmonic Mean based on test case fault (HM_{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,

$$HMtf = \frac{m}{\frac{1}{TF1} + \frac{1}{TF2} + \cdots + \frac{1}{TFn}} \tag{2}$$

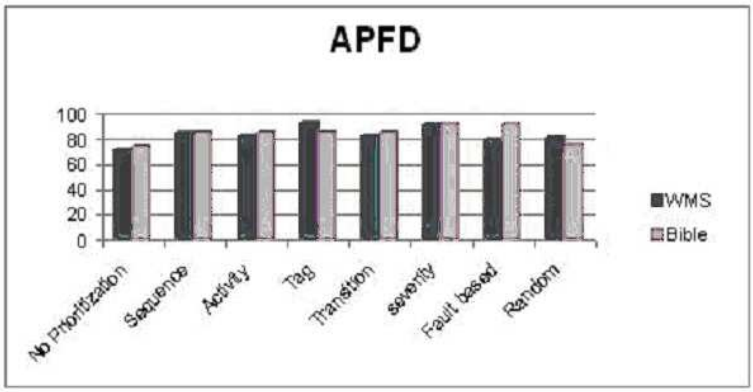

Fig. 9 *Comparison of APFD Values.*

Where T be 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. The HM value is calculated for the various applications is shown in Tab. IV below.

Application	No prioritization	Sequence	Activity	Tag	Transition	Severity	Fault based	Random
Weather monitoring system	5.76	6	6.26	8.57	6.26	11.25	5	4.2
Bible	6.14	7.65	7.65	7.65	7.65	12	12	4.1

Tab. IV *Harmonic mean (HM_{tf}) value for different web services.*

Fig. 10 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.

6.6 Harmonic Mean Sequence Invocation (HM_{si})

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,

$$HMsi = \frac{m}{\frac{1}{si1} + \frac{1}{si2} + \cdots \frac{1}{sin}} \tag{3}$$

Where T be a test suite consisting of n test cases and F be a set of m faults revealed by T. Si be the number of invocation of web services in the reordered test suite S of T that reveals fault i. The $HM(si)$ value is calculated for the various applications is shown in below Tab. V.

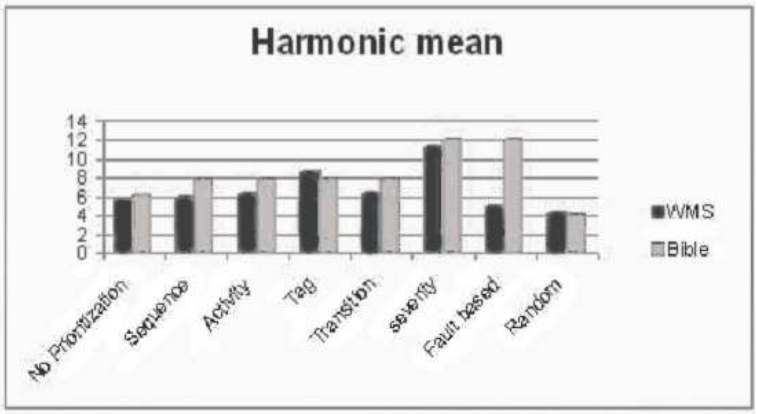

Fig. 10 *Comparison of harmonic mean for web services.*

Application	No prioritization	Sequence	Activity	Tag	Transition	*Severity*	Fault based	Random
WMS	4.36	12	7.2	30	6.55	30	4.36	3.06
Bible	7.41	8	8	8	8	24	24	0.68

Tab. V *Harmonic mean sequence invocation ($HMsi$) value for different web services.*

Fig. 11 shows HMsi values for all applications for proposed techniques. It is clear from the figure that the severity based technique has the maximum value of 30 compared to other techniques.

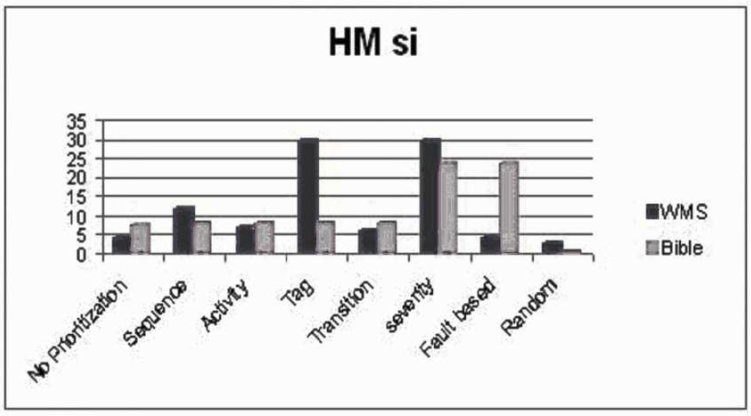

Fig. 11 *Comparison of harmonic mean sequence invocation for web services.*

Threats to Validity

The construct validity of our experiment relates to the metrics used to evaluate the effectiveness of test case prioritization. We use the metrics $APFD$ in the experiment. Although normally knowing the faults exposed by a test case in advance is impractical, and hence an $APFD$ value cannot be estimated before testing has been done.

However, $APFD$ can be used as a measure to show the feedback of prioritization techniques when testing has finished. The external validity is whether the experiment can be generalized. We use WS-BPEL applications as subjects. They are a representative kind of service-oriented business application. Our experiments can be conducted using other service-oriented applications that use XPath queries and WSDL specifications. We will find more such applications to evaluate our techniques.

7. Conclusion

In this paper we have proposed partially automated model for testing composite web services where the logical sequence flow between the web services (semantics) was included through OWL-S. This model also included the various techniques for prioritizing techniques generated for the composite web service. Experiments were conducted and the evaluation of techniques were performed based on $APFD$, HM_{tf}, $HMsi$ and 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 index such as the resource, Cost based and the like. The Composition is done with the concept BPEL (Business Process Execution Language) for improving as per the industry standard.

References

[1] Askarunisa A., Abirami A. M., Madhan Mohan: A test case reduction method for semantic based web services, ICCCNT, July 2010.

[2] 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.

[3] Rothermel G., Untch R., Chu C., Harrold M.: Test Case Prioritization, IEEE Transactions on Software Engineering, **27**, October 2001, pp. 929-948.

[4] Gannod G. C., Brodie R. J., Timm J. T. E.: Foundations for Specifying OWL-S groundings, Int. J. Business Process Integration and Management, **2**, I, 2007.

[5] GraphViz `http://www.research.att.com/sw/tools/graphviz/download.html`

[6] II Woong Kim, 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, **39**, 6, November 2009.

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

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

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

[10] Paolucci M., Wanger M.: Grounding OWL-S in WSDLS, IEEE International Conference on Web Services, 2006.

[11] OWL-S Editor: `http://sta_.um.edu.mt/cabe2/supervising/undergraduate/owlseditFYP/download.html`

[12] Protégé tool Tutorial: `www.Protege.stanford.edu/doc.users.html`

[13] Protégé tool: http://protege.stanford.edu/download.html

[14] Kavitha R., Sureshkumar N.: Test Case Prioritization for Regression Testing based on Severity of Fault, (IJCSE) International Journal on Computer Science and Engineering, **02**, 05, 2010, pp. 1462-1466.

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

[16] Noikajana S., Suwannasart T.: An Improved Test Cases Generation Method for Web Services Testing from WSDL-S and OCL with Pair-Wise Testing Technique, 33rd Annual IEEE International Computer Software and Applications Conference, 2009.

[17] Web inject tool: `www.webinject.org`

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

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

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

[21] Yongbo Wang, Xiaoying Bai, Juanzi Li, Ruobo Huang: Ontology based test case generation for Testing Web Services, 8th International Symposium on Autonomous decentralized Systems, 2007.