ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/261039283

A tool supported approach to perform efficient regression testing of web services

Conference Paper · September 2013

DOI: 10.1109/MESOCA.2013.6632734

CITATIONS
12

READS
153

2 authors:

Animesh Chaturvedi
Indian Institute of Information Technology, Design and Manufacturing Jabalpur
17 PUBLICATIONS **112** CITATIONS
SEE PROFILE

Atul Gupta
PDPM Indian Institute of Information Technology, Design and Manufacturing Jaba…
56 PUBLICATIONS **860** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

3rd International Conference on Computer Vision and Image Processing (CVIP 2018) View project

Security of Web Applications View project

All content following this page was uploaded by Animesh Chaturvedi on 05 June 2015.

The user has requested enhancement of the downloaded file.

A Tool Supported Approach to Perform Efficient Regression Testing of Web Services

Animesh Chaturvedi
Department of Computer Science Engineering
Indian Institute of Information Technology, Design and Manufacturing Jabalpur, IIITDM-J
Jabalpur, India
animesh@iiitdmj.ac.in

Atul Gupta
Department of Computer Science Engineering
Indian Institute of Information Technology, Design and Manufacturing Jabalpur, IIITDM-J
Jabalpur, India
atul@iiitdmj.ac.in

Abstract— **In this paper, we present a tool supported approach to perform efficient regression testing of web services. Functional and non-functional web service testing is done with the help of WSDL parsing and regression testing is performed by identifying the changes made thereafter. We identify, categorize, and capture the web service regression testing needs into three different categories, namely, changes in WSDL, changes in code, and selective re-testing of web service operations. To capture above three changes we proposed three intermediate forms of WSDL, namely, Difference WSDL (DWSDL), Unit WSDL (UWSDL), and Reduced WSDL (RWSDL), respectively. These intermediate forms of WSDLs are then combined to form Combined WSDL (CWSDL) which is further used for regression testing of the web service. This approach is prototyped as a tool, named as Automatic Web Service Change Management (AWSCM), which helps in performing the efficient regression testing of web services by selecting the relevant test cases to constructing reduced test suite from the old test suite file of SoapUI. The reduction in the effort for regression testing of web service is estimated by two proposed cost metrics. We present three case studies demonstrating the applicability of the proposed tool for the real world projects.**

Keywords- Web service, web service testing, WSDL, web service change management, regression testing.

I. INTRODUCTION

Web service (WS) solves the problem of interoperable machine to machine communication over internet. WSDL is used as an interface for the communication between two different languages and machines. A web service can be a standalone (or a part of) web application that can be independently developed, deployed and tested. Web services are backbone of current internet world. Web services can be implemented in variety of ways like using object-oriented programming in Java or C++, or using procedural languages like C. Many new technology domains like Cloud computing and Service oriented computing typically rely upon WS technology domain.

As with any software solution, testing is the most important form of validating the software. WS testing is performed by generating automated test cases for operations by parsing the WSDL of the WS [1, 2]. WS Testing involves communications between the WS and clients with soap requests and responses. White box WS testing can be applied by traditional methods of regression testing (i.e. control and data flow graph). Black box WS testing can be done by WSDL with software tools like SoapUI [25] and JMeter [26].

Regression Testing of Web Service (RTWS) can be optimized by doing selective regression testing of modified and inserted portion only with the guaranteed that new and old code would conform to the changes in requirements [3]. This avoids the costly construction of unnecessary test cases and unproductive re-run of the useless test cases which leading to reduced regression testing efforts [4, 5].

This paper reiterates an earlier observation of selective regression testing in the context of WS, which is more cost effective than testing the entire system. The proposed AWSCM tool which generates smaller WSDL interface descriptions to creates Reduce Regression Test Suite (RRTS) for testing purposes. WS automated functionality testing for operations can be done with tools like SoapUI, WebInject and Jmeter. The code implementing the web services can be maintained with the traditional regression testing techniques [6, 7]. The main contribution of this paper is that it captures the changes at WSDL and code level, and maps these changes to smaller CWSDL. Operations in CWSDL are further mapped to their respective test case to do efficient RTWS by facilitating the standard WSDL parsing techniques.

The rest of the paper is as follows. Section II describes the related work. Section III, we discuss our approach of change identification and porting them to intermediate forms of WSDLs. Section IV presents the details of performing efficient RTWS. Section V presents the metrics for estimating the reduction in the cost due to the efficient regression testing performed by the tool. We present three case studies highlighting applicability of the proposed approach to perform RTWS in real settings in Section VI. Finally, we draw our conclusions and future work in Section VII.

II. RELATED WORK

Regression testing aims at testing the changes made to the code and ensuring that the changes are not causing any side-effects. One of the ways this can be ensured by re-running all the valid old test cases in old test suite file, which however, can be prohibitively costly. A better approach can be to detect the changed portion of the code and select test cases that mapped to changed and unchanged portions of the code. David Binkley described program that generates reduced test cases by identifies components that need tests after getting differences and similarities between the old and new programs [4]. Rothermel and Harrold used CFG of program and its

978-1-4673-4889-8/13/$31.00 ©2013 IEEE

modified version to select tests cases that execute changed code from the original test suite [6]. Romano and Pinzger proposed a tool called WSDLDiff to extract fine-grained changes from subsequent versions of WSDL's of WS [8], similar work in reducing cost in RTWS is in [5].

W. T. Tsai et al. described four ways in which the WSDL can be extended to facilitate WS testing in [2]. X. Bai et al. proposed a WSDL based automatic test case generation approach for WS testing [1] is done by parsing WSDL [1] and the same forms the basis of automate WS testing tools like SoapUI and Jmeter. In [9] regression testing of WS applications with the safe RTS, FSA techniques is based on white box but they are not use WSDL for test case generation like in testing tool WebInject, SoapUI, Jmeter. A detailed description of safe regression test selection technique for web services is presented in [7], complete description of RTWS in [10], and model based RTWS is presented in [11]. Sana Azzam et al. [12], presented an analysis on the basis of various testing approaches of different WS to compare automated WS tools (SoapUI, PushToTest, Jmeter, WebInject); but they did not perform validations of the tools on the basis of regression testing. M Ruth et al. [13] described a code transformation based approach that transforms the service code into a local program for applying a safe RTS technique in an end-to-end manner. Hema and Myra presented an industrial case study of a regression testing approach to improve test effectiveness and efficiency in SaaS by prioritize the tests to improve time to detection of faults in the modified system [14]. Bixin Li et al. proposed an approach to select test cases for identifying the changes by performing control flow analysis of different versions of BPEL composite service where these changes are involved [15]. M. Bruno et al. proposed the use of test cases as a form of contract between the provider and users of a service by repeatedly run a test suite [16]. Drawback of regression testing of WS in previous works is that they considered either white box or black box without considering change impact on WSDL. Moreover, some build their own WS testing tool instead of using standard tool to convey their approach of RTWS.

A. Pasala et al. proposed a prototype tool for RTS selection based on analyzing the dynamic behavior of the application [3]. J. Zheng et al. presented I-BACCI process for change identification and then performing regression test selection, conducted in scenario of black-box [17].

Thirumaran. M et al. in [18] proposed a dynamic business logic metric and algorithms for change factors evaluation helps in reducing the change management issue in WS. Xumin L., Athman B. and others in [19, 20] proposed methodology for handling Top down changes in Long-term Composed Services (LCS), by focusing on changes in replacement or addition of a WS.

III. DIFFRENCE, UNIT, REDUCE & COMBINED WSDL

Web service changes are the reason to perform monitoring and retesting of WS. To reduce effort of monitoring and retesting, an automated engineering is required for identifying changes (diff). Changes in any document, program, product and WS could be in three ways: deleted, inserted or modified for achieving these requirement; this paper proposed four types of WSDL's i.e. (D/U/R/C) WSDL. Firstly, DWSDL can be defined as WSDL constructed from the diff between two versions of WSDL, changes at WSDL's level are mapped by some standard automated change identifying tool. For example operation3 in Fig. 1, is an inserted operation between two subsequent versions of WS. Secondly, RWSDL can be defined as WSDL constructed by user selective operations of WSDL. Thirdly, UWSDL can be defined as WSDL which is constructed with the operation undergone through change in code level. Fourthly, CWSDL can be defined as WSDL which is constructed with the operations in (D/U/R) WSDL such that it contains only unique and non-redundant operations (i.e. remove redundant operations in (D/U/R) WSDL).

Automatically gathering changes at WSDL and code (logical statement) level helps in effort reduction in RTWS. If the changes to the WS are at the code level then changes are either reflected or not reflect on the WSDL in Table I. First if change reflected on WSDL, then they can be captured by (D/R) WSDL and they are especially helpful whenever tester is exposed only to the WSDL's and do not have any details of WS code. Limitation with (D/R) WSDL is that they do not identify the changes at logical code level. But still, tester requires performing code change based regression testing which can be done by using UWSDL. Second if coding changes are done to the WS which are not reflected on the WSDL, then they can be captured by UWSDL. By using UWSDL constructed automatically by AWSCM tester can perform code based RTWS efficiently.

TABLE I. CHANGES CAPTURED BY DWSDL, RWSDL, UWSDL

Scope of changes handling by utilities	
Utilities	***Capture changes WSDL or at code level***
DWSDL	WSDL, operation deletion, insertion or I/O modification
RWSDL	WSDL, select operations on WSDL to test
UWSDL	Code of operation changes are captured

Construction of (D/U/R) WSDL with AWSCM is done by gathering changes with semantic textual difference using JDIff [21], Predic8 [22] and Regex [23] utilities and libraries. First, DWSDL utility of AWSCM takes input as new and old WSDL's to gather changes between two versions of WS. Using these changed operations, AWSCM semantically constructs DWSDL. Second, UWSDL utility of AWSCM takes three inputs: new, old code file of operations and new WSDL. Tool semantically constructs UWSDL according to semantics of new WSDL with those operations which are gone through the change in the logical statement level of code. Third, RWSDL utility of AWSCM takes any WSDL, gather its operations and then offer user to selected operation that is required in performing RTWS. Using these selected operations, AWSCM semantically constructs RWSDL.

Any WSDL must be in proper structure and compliance to WSDL standards for maintaining communication with the WS logic. While constructing (D/U/R/C) WSDL by AWSCM, we considers semantic, syntactical and data correctness i.e. standard syntax (tags) used by the WSDL of WS project, and having same data for the port, service, binding and schema (input/outputs of operation) as in the

inserted new version of WSDL. (D/U/R/C) WSDL must have correct XSD types, Message part, Operation, Port type, Binding, Port and Service according to input WSDL.

If tester can automatically get the information about the changes in the particular operation of WS then tester can test that particular operation to validate requirements. Tester can generate RRTS file from two inputs, first old Test suite file, second (D/U/R/C) WSDL. These RRTS contained test cases only for those operations which were in the (D/U/R/C) WSDL. Description of regression testing using (D/U/R/C) WSDL's with the help of testing tools SoapUI is given in the next section.

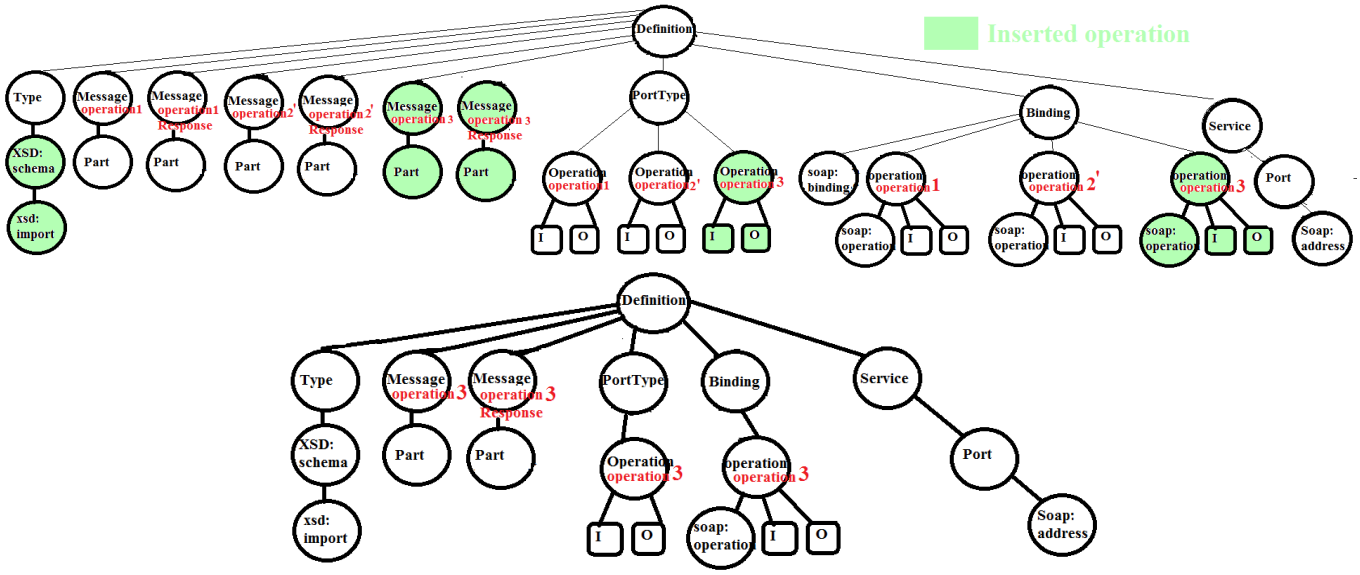

Figure 1. Construction of WSDL for operation3 from new WSDL with operation3

IV. Regression Testing of Web Service With AWSCM

Tester can use (D/U/R/C) WSDL in two ways to perform efficient regression testing. Firstly, apply (D/U/R/C) WSDL one by one to the Jmeter or SoapUI to generate testing templates to write new test cases for inserted operations. With these templates tester can perform testing of inserted operation. Secondly, user can retrieve reduced test case from the old test suite file on the basis of (D/U/R/C) WSDL. This can be done by manipulating the information inside source code of operations in (D/U/R/C) WSDL to select the test steps from the old test cases to perform regression testing.

Changes in the WS could occur in three ways i.e. deletion, insertion and modification. Firstly, deletion on an operation means that particular operation will not be further in the next version of WS; such operations should not require for RTWS. Secondly, insertion could be take place by inserting a new operation; such operations can be identified from WSDL change between two versions and should be considered for RTWS, for this DWSDL and RWSDL is used in Fig. 4. Thirdly, modification could occur in two levels at the WSDL level and at the code level. First, modification at WSDL level means change in XSD i.e. when for a particular operation only input or output or both is changed, in this DWSDL and RWSDL is used in Fig. 2 (a). Second, if modification is at the code level, then this UWSDL is used in Fig. 2 (a). Change impact analysis to gather the changes and map it into the test suite file in terms of test steps inside test case is done within UWSDL utility. Impact analysis is done on the basis of commonly known control and data flow analysis. In AWSCM change identified in the code of the operation in two ways by JDiff for getting changes line by line and regular expression for data flow analysis. Firstly indentifying line by line changes to find the changes between files with the same name inside folder named as New and Old. Folder New and Old contains files which are named on the name of operations, and each file contains code of operation on which it is named. Apply JDiff between the entire operation file in New and Old folder, find file (operations) which are changed. Gather mapping for test data or step inside the test cases according to the changes. Use those test data or step inside with respective test case of the operation for construction of RRTS. Secondly, test data flow analysis by gathering input/output parameters which are affected during changes in the code, which must be re-tested again by rerunning test data or steps. This is done by gathering those parameters by utility of regular expression (AWSCM used "java.util.Regex"). After identifying changes at the code level, tester could construct UWSDL and RRTS according to the changes inside the operation.

For operation in CWSDL we can construct Combined RRTS (CRRTS) which contains all the unique test cases i.e. eliminate the redundant test case. Hence tester will have small number of useful test cases to perform cost effective RTWS.

V. Cost Reduction Estimation Metrics

Although it is tough to calculate exact effort reduction, therefore authors proposed two ways for approx calculation of cost reduction. Cost reduction estimation metric in the context of RTWS is a proposed technique to find out approx effort reduction in RTWS by taking change information's of WS. First, in terms of average change in number of lines and second, cost reduction in terms of operation count. Table III in Section VI finds effort reduction from ratio of number of

line changed in WSDL and code with total lines of WSDL and code new version. Whereas, Table IV simply finds the ratio between number of operations changed in WS with total number of operations in new WSDL. Where, LW stands for Lines of WSDL, Op stands for Operation and LoC stands for Lines of Code.

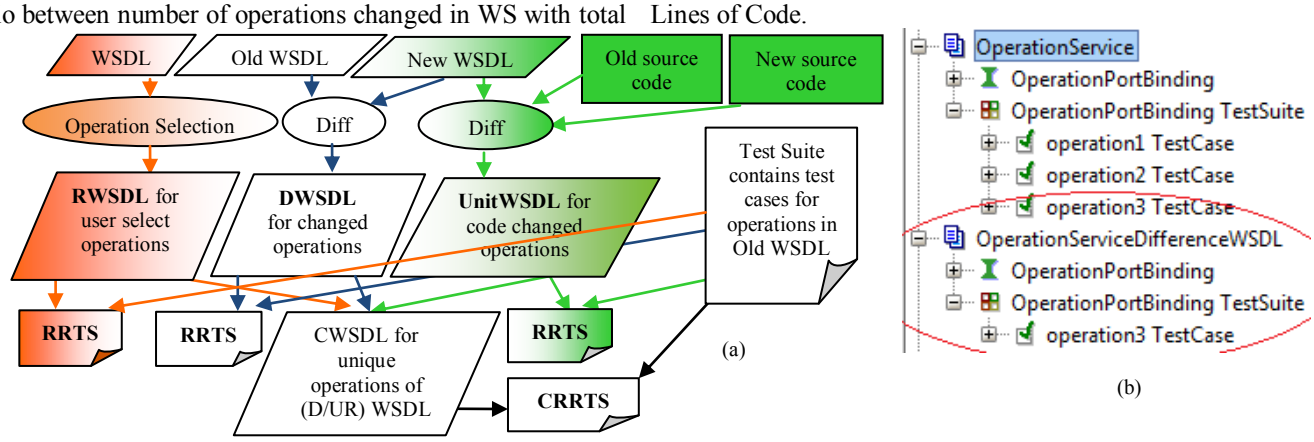

Figure 2. (a) Three RRTS are constructed by automated selection of Test Cases of old Test Suite for the operation in the RWSDL, DWSDL and UWSDL respectively. (b) Simple RRTS in SoapUI for example in fig. 3

VI. CASE STUDIES

In this section, we present AWSCM supported three case studies on WSDL based WS projects, namely, Eucalyptus CC (cloud controller) cc_wsdl is available on Github, Amazon Web service, SaaS WS (in different versions) and some other WS projects. Simplest case study for Fig. 3 was conducted using the proposed tool AWSCM for constructing reduced test cases from old test suite shown in Fig. 2 (b). The WS projects along with developed intermediate forms of WSDLs in our case studies are shown in Table II.

TABLE II. WS PROJECTS AND CONSTRUCTED WSDLS

Case Study WS Projects	CWSDL		
	DWSDL	***RWSDL***	***UWSDL***
Eucalyptus	Y	Y	Y
SaaS	Y	Y	Y
Amazon WS	Different versions of WSDL is not available	Y	Code is not available
Bible WS		Y	
Currency Conversion WS		Y	
Weather WS		Y	

Eucalyptus and Amazon WS are easily available on internet, and they are the evolving project and continuously changing to maintain the confounding factor of source code may available or may not available. Eucalyptus_CC is used in all (D/U/R/C) WSDL constructions and depending upon the availability of Amazon WS WSDL only, RWSDL is constructed.

First case study is conducted over two versions in branch (/maint/master 3.1, master) of cc_wsdl and codes (cluster/handlers.c) selected from Eucalyptus repository at Github [24]. The application of the DWSDL to the SoapUI generated reduced test case templates for inserted operations, namely, DescribeSensors, BundleRestartInstance, ModifyNode and MigrateInstance as shown in Fig 3 (a). We can put additional test data in those templates to test newly inserted operation for the eucalyptus. Selective RTWS is performed by selecting the targeted operations to construct RWSDL from WSDL of Amazon WS as shown in Fig. 5 and WSDL of EucalyptusCC. RRTS file is generated from the old test suite file for operation in RWSDL in Fig. 3 (b).

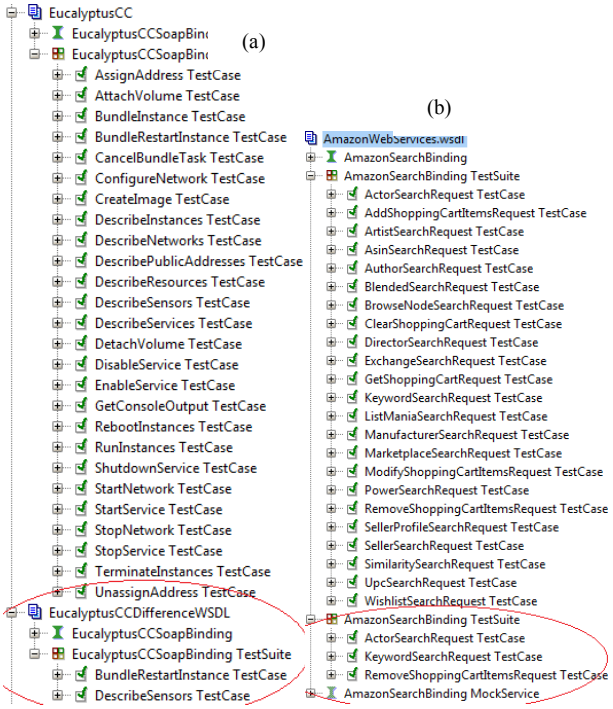

Figure 3. Snapshot of SoapUI showing old test suite and RRTS (a) Eucalyptus_CC, (b) Amazon consumer WSDL

The second case study involved, which SaaS_1 contains operations, namely, index, reading, searching, editFile. The operation editFile was changed at the code level to the make second version of SaaS i.e SaaS_2. Construction of UWSDL from WSDL and the source code of the two versions (SaaS_1 and SaaS_2) were given as input to AWSCM. The AWSCM constructed the RRTS file (which contains test cases of only editFile operation as shown in fig. 4 (a)) from UWSDL and the old test suite file given as input. Similarly the construction of DWSDL with consecutive version of WSDL's for WS SaaS was followed. Similarly for construction of RWSDL

user selected operation of new WSDL to generate RRTS from old test suite.

In the third case study, test data reduction for four dynamic web services, namely, currency conversion, Global weather, Sunset service, Bible Web services. Using the tool, we constructed RWSDL which further used to construct RRTS. The RRTS file is constructed according to the user selected scenarios which contain test data inside test cases similar to old test suite in Fig. 4. AWSCM also does the code analysis of Saas and Eucalyptus_CC (handler.c at Githhub) to perform data flow analysis. In this feature AWSCM gathers affected parameter and based on these parameter tool had constructed UWSDL with these operation having those parameters. Change impact analysis using control and data dependencies flow is done to find out which parameters are affected. Gathered impacts are mapped to the test steps which are inside test cases for a particular operation of the test suite.

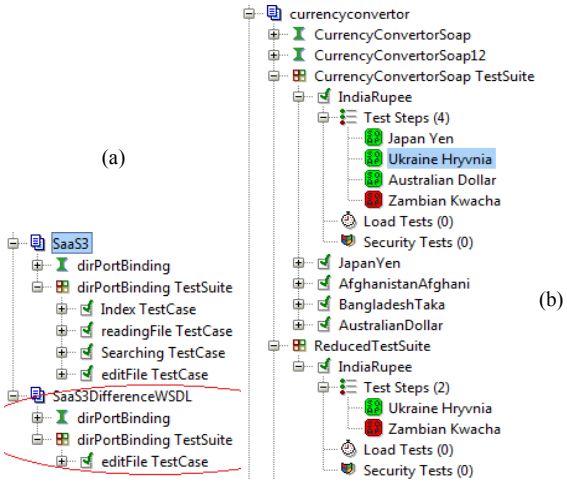

Figure 4. SoapUI snapshot of RRTS (a) for changed operation "editFile" in SaaS, (b) test step of "India".

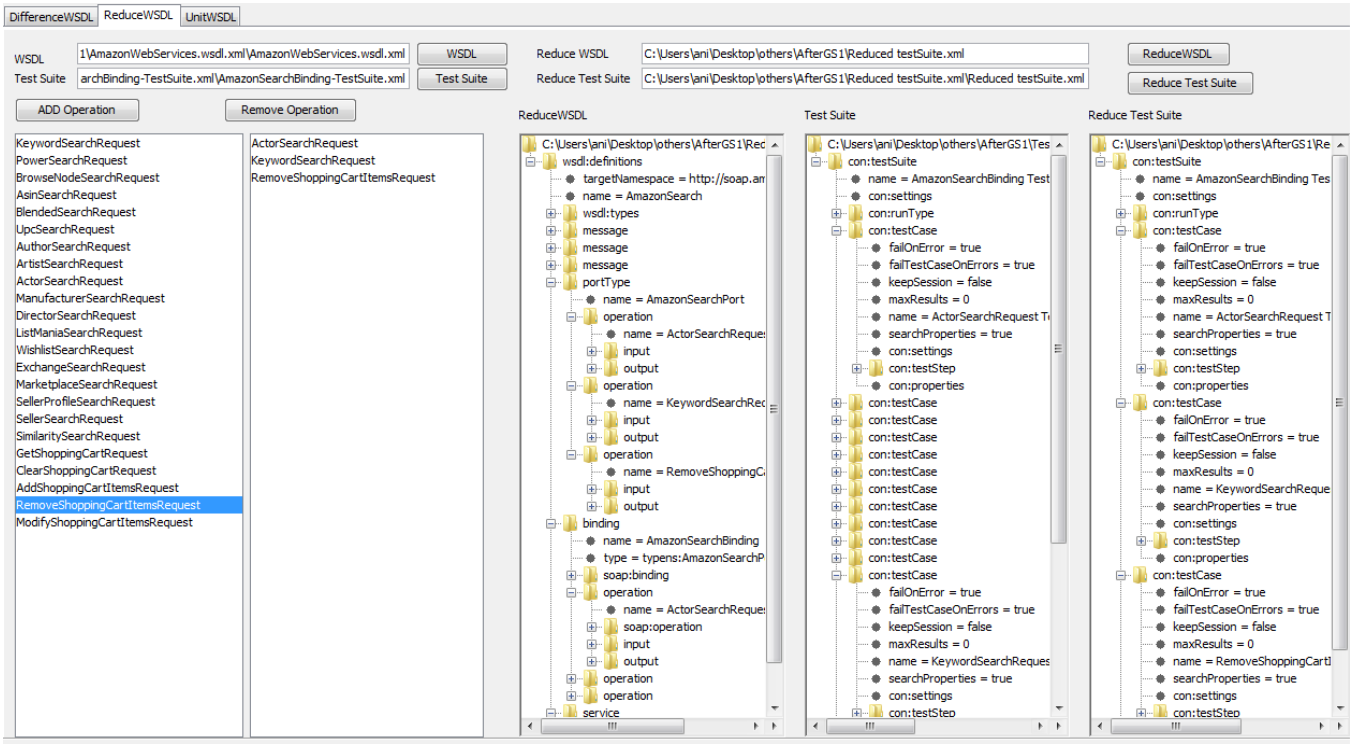

Figure 5. ReduceWSDL utility of AWSCM for Amazon WSDL.

TABLE III. AVERAGE CHANGE IN NUMBER OF LINES COST METRIC

Average change in number of lines cost metric, declaration of variable, data (approx) and calculation				
Variables	***Quantity***	***Unit***	***SaaS***	***EucalyptusCC***
Number (No.) of lines in WSDL_1	L1	LW	77	1712
No. of lines in WSDL_2	L2	LW	115	1492
Change in line of WSDL's	C = L1 - L2	LW	38	220
Average number of operation / line of WSDL_1, WSDL_2	X1, X2, Xav = (X1+X2)/2	Op / LW	26, 23, 24.5	62, 61, 61.5
Average Line of Code (LoC) per operation	Y1, Y2, Yav = (Y1+Y2)/2	LoC / Op	40,56, 48	216, 291, 253.5
No. of LoC for every operations	C * Xav	Op	38 * 24.5	220 * 61.5
No. of Line of Code (to be tested) V'	C * Xav * Yav	LoC	38*24.5*48	117*61.5*253.5
No. of Line of Code in V_2	L2 * X2 * Y2	LoC	115*23*56	1712*61*291
Effort required	V' / V_2		44688 / 148120	1824059 / 30389712
Percentage effort required	(V' / V_2) * 100	%	0.3017 * 100	0.0600 *100
Percentage effort reduction	100 – (V' / V_2) * 100	%	100 – 30.17	100 – 6.00

TABLE IV. OPERATION COUNT COST METRIC

Operation count cost metric				
Variables	***Quantity***	***Unit***	***SaaS***	***Eucalyptus CC***
No. of operations in WSDL_1	X	Op	3	24
No. of operations in WSDL_2	Y	Op	4	28
No. of operations in DWSDL	Z	Op	1	4
No. of operations in DWSDL	Y-Z	Op	1	4
Percentage reduction in operation by DWSDL	((Y-Z)/Y) *100	Op	75%	85.7%

Comparing the two versions of SaaS, AWSCM got reduction of testing effort of 70 to 75 % approx i.e. 30 to 25 % approx of changes for RTWS. Similarly, two versions of Eucalyptus WSDL, AWSCM got 85 to 95% approx reduction in testing effort for RTWS.

VII. CONCLUSION AND FUTURE WORK

In this paper, we proposed a tool-supported approach to find out exactly which operations are changed and how to use only these operations for reduced regression testing. The tool AWSCM constructs three types of intermediate WSDL, namely, DWSDL, RWSDL and UWSDL to generate three RRTS respectively. CWSDL is constructed by combining the unique operations in (D/U/R) WSDL; efficient RTWS is performed by CRRTS generated from CWSDL. To compute the effort reduction, we have proposed two cost estimation metrics based on the change ratio between number of operations or lines in new version WSDL and (D/U/R/C) WSDL. Further, we plan to extend UWSDL to capture changes in complex interactions between WS operations.

REFERENCES

[1] Xiaoying Bai, Wenli Dong, W. Tek Tsai and Yinong Chen, "WSDL-Based Automatic Test Case Generation for Web Services Testing," *Proc. Int. Workshop on Service-Oriented System Engineering* on IEEE, 2005, pp. 207-212.

[2] Wei-Tek Tsai, Ray Paul, Yamin Wang, Chun Fan, and Dong Wang, "Extending WSDL to Facilitate Web Services Testing," *Proc 7th IEEE Int. Symposium on High Assurance Systems Engineering (HASE'02)*, 2002, pp. 1-2.

[3] Anjaneyulu Pasala et. al., "Selection of Regression Test Suite to Validate Software Applications upon Deployment of Upgrades," *Proc. 19th Australian Conference on Software Engineering* on IEEE, 2008, pp. 130-138.

[4] David Binkley, "Semantics Guided Regression Test Cost Reduction," *IEEE Trans. on Software Engineering*, vol. 23, no. 8, August 1997, pp. 498-516.

[5] Animesh Chaturvedi "Recuding cost in Regression Testing of Web Service," *Proc. Sixth CSI International Conference on Software Engineering (CONSEG)* on IEEE, 2012, pp. 1-9.

[6] Gregg. Rothermel and Mary Jean Harrold, "A safe, efficient regression test selection technique," *ACM Trans. Software Eng. Methodol*. Vol. 6, No. 2, April 1997, pp. 173-210.

[7] M. Ruth and S. Tu, "A Safe Regression Test Selection Technique for Web Services," *Second Int. Conference on Internet and Web Applications and Services* on IEEE, 2007.

[8] Daniele Romano and Martin Pinzger, "Analyzing the Evolution of Web Services using Fine-Grained Changes," *Proc. of 19th IEEE Int. Conference on Web Services*, 2012, pp. 392-399.

[9] Abbas Tarhini, Hacène Fouchal and Nashat Mansour, "Regression Testing Web Services-based Applications," *Proc. IEEE Int Confrence Computer Systems and Applications*, March 8, 2006, pp. 163 – 170.

[10] M. Di Penta, Marcello Bruno, Gianpiero Esposito, Valentina Mazza, and Gerardo Canfora., "Web Services Regression Testing," *Test and Analysis of web Services*, 7 May, 2007, pp. 205-234.

[11] Tamim Ahmed Khan and Reiko Heckel, "A Methodology for Model-Based Regression Testing of Web Services," *Proc. IEEE Testing: Academic and Industrial Conference - Practice and Research Techniques*, IEEE Computer Society, 2009, pp. 123-124.

[12] Sana Azzam, M. Naji Al-Kabi and Izzat Alsmadi "Web Services Testing Challenges and Approaches," *ICCIT*, 2012, pp. 291-296.

[13] M. Ruth, F. Lin and S. Tu, "Applying Safe Regression Test Selection Techniques to Java Web Services," *Int. Journal of Web Services Practices*, vol.2, No.1-2, 2006, pp. 1-10.

[14] Hema Srikanth and Myra B. Cohen, "Regression Testing in Software as a Service: An Industrial Case Study," *Proc. 27th IEEE Int. Confrence on Software Maintenance (ICSM)*, 2011, pp. 372-381.

[15] Bixin Li, Dong Qiu, Hareton Leung and Di Wang, "Automatic test case selection for regression testing of composite service based on extensible BPEL flow graph," *Journal of Systems and Software* vol. 85, 2012, pp. 1300–1324.

[16] Marcello Bruno, Gerardo Canfora and M. Di Penta, "Regression Testing ofWeb Services," *RCOST Technical Report*, April 2005.

[17] Jiang Zheng, Laurie Williams, Brian Robinson and Karen Smiley "Regression Test Selection for Black-box Dynamic Link Library Components," *Proc. 2nd Int. Workshop on Incorporating COTS Software into Software Systems: Tools and Techniques*. IEEE Computer Society, 2007.

[18] Thirumaran. M et al, "Evaluation of Change Factors for Web Service Change Management," *2nd Int. Confrence on Communication, Computing & Security*, 2012, pp. 163-170.

[19] Xumin Liu, AthmanBouguettaya, Qi Yu and Zaki Malik, "Efficient change management in long-term composed services," *Journal of Service Oriented Computing and Applications*, Springer-Verlag London Limited, 2010, pp. 87-103.

[20] Xumin Liu, Athman Bouguettaya, Jemma Wu, Li Zhou, "Ev-LCS: A System for the Evolution of Long-Term Composed Services," *IEEE Transactions on Services Computing*, vol. 6, no. 1, 2013, pp. 102-115.

[21] Apiwattanapong T, Orso A, and Harrold M J, "JDiff: A Differencing Technique and Tool for Object-Oriented Programs," *Journal of Automated Software Eng.*, Vol. 14, No. 1, March 2007, pp 3-36.

[22] Membrane SOA Model," April 7, 2013; http://membrane-soa.org/soa-model/.

[23] "Pakage java.util.regex," April 7, 2013; http://docs.oracle.com/javase/6/docs/api/java/util/regex/package-summary.html.

[24] "Eucalyptus," 6,July,2013; https://github.com/eucalyptus/eucalyptus.

[25] "SoapUI," 24, July, 2013; http://www.soapui.org/.

[26] "JMeter," 24, July, 2013; http://jmeter.apache.org/.

View publication stats