Subset WSDL to access Subset Service for Analysis

Animesh Chaturvedi
Indian Institute of Information Technology, Design and Manufacturing - Jabalpur
animesh.chaturvedi88@gmail.com

***Abstract*— Service analyzer requires automated approach to access Subset Service, so that cost can be reduced by organizing the test scenarios. Emphasis of analyzer is to access and handle subset of code. This paper proposed a conceptual model that access Subset Service in two steps. First, slicing the Web service based on Subset WSDL (SWSDL) to access Subset Service. Second, the Subset Service is categorized into five layers. This improves and optimizes cost of Web service analysis by slicing and layering the Service. We conducted case studies for few Service projects. SWSDLs are used for Operationalized and Parameterized Web service analysis.**

Keywords*— *Web Service, WSDL, regression testing, top down development, change impact analysis, slicing and SOAP.

I. INTRODUCTION

A Web Service (WS) is a composable unit of web application, which can interact with other applications using operations specified in an XML based interface called Web Service Description Language (WSDL). Operation of WS is similar to a method signature of functional programming. WS can also be independently developed and deployed as a standalone web application.

Normally, functional and non-functional testing of WS is performed by testing the operations of WSDL. In general, communication with SOAP message via WSDL is used to perform automated WS analysis. Analysis techniques can be used in change impact analysis, top down development, and legacy/contract monitoring. WS analysis can be performed to identifying traceability of the affected portions of WS code and WSDL. Analysis cost can be optimized by properly identifying all the affected portions of the WS. Thereafter these identified portions are used to select old test case.

Web Service Analysis (WSA) technique is required for WS legacy contracts validation, cost optimization and QoS. WSA techniques are required for regression testing and top down development. Regression testing requires change impact analysis for selective verifying the requirement changes. Top down development requires analysis of functionality or operations. Generally, WSA can be done by using information of WSDL, XSD, code and its composition.

State of art is add-on by categorizing various types of necessary WSA. This paper described WSA by accessing subset service using subset WSDL generated by a proposed prototype tool. Selective code portions are used to slice the WS in subset to reduce analysis efforts. Early paper by the author are [7] on change impact analysis based regression testing of WS. The papers had given mapping of changes in the WSDLs and WS code to the test suite will be helpful in regression testing efficiently. This paper proposed an approach to construct Subset WSDL (SWSDL) to access Subset Service for categorized analysis.

II. RELATED WORK

Bai et al. in [1] proposed WSDL based automatic test case generation for WS testing. Similarly, we can do WSDL based WSA. M Ruth in [2] described a safe regression test selection technique for WS with control flow and call graph. Romano & Pinzger proposed a tool called WSDLDiff [4] to extract fine-grained changes from subsequent versions of WSDL. El Bouhissi and Malki in [3] proposed a tool based on WS Modeling Ontology (WSMO). WSMO and WSDLDiff gather the selection & change in WSDL. We can further use WSDLDiff and WSMO information to construct Subset WSDLs. Lui, Bouguettaya and others in [6] proposed methodology for handling top down changes in Long-term Composed Services (LCS), by focusing on WS changes in replacement or addition.

III. SUBSET WSDL (SWSDL) TO ACCESS SUBSET SERVICE

Generally, WS operations are loosely coupled so each operation can be separately analyzed for their business logic, call request and response. A WS operation is similar to a method or function in a traditional programming language. Client and WS code communication is done with SOAP message between, WSDL as interface defines operation. We can use WS communication flow to identify all affected portions. We can slice the WS code vertically for WSA to execute the WS operation wise as shown in figure 1 (a).

Normally, WSDL is used for automated accessing, manipulation and analysis of WS code. This automation can be used for change imapact analysis based regression testing, analysis based top down development, and legacy/contract based WS monitoring. Normally, change imapact analysis is required for regression testing. We will construct SWSDLs using info of change imapact analysis on WSDL and WS code. As we know, analysis of operations is required for top down development. We will construct SWSDL using info of automatic operational analysis from the input WSDL and WS code. Using SWSDL, we can perform efficient regression testing and top down development of WS.

Automatically gathered operations are used to semantically construct SWSDL. SWSDL will help in many domains/fields of WSA. SWSDL constructed with 'requiredOperations' such that these operations are required to be analyzed for given input WSDL. Extract all required data and part (message, XSD schema, port, binding, service etc) semantically from the input WSDL. Arrange all the parts semantically according to the WSDL standards and input WSDL, such that a client can

978-1-4799-4093-6/14 $31.00 © 2014 IEEE
DOI 10.1109/CloudCom.2014.149

properly communicate with the WS code via SWSDL. Construction of SWSDL is three-step process.

A. Gather operations ('requiredOperations') that are present in '*wsdl*' required for analysis of WS.

B. Construct the various parts of WSDL (Definition, XSD, Message, Port, Binding and Service) for the operations 'requiredOperations' gathered in first step. Definition and Service are constant (i.e. independent of 'requriedOperations'), whereas others are dependent on the operations. Steps to gather various parts of WSDL:

1) constructStartDefinition: cStartDef is constructed with extraction of string element <?xml….?>. Thereafter, append extraction of string with tag <definitions…> for Definition (depending on WSDL).

2) constructXSD, cXSD is constructed with the extraction of string for all the Schemas of 'requiredOperations' in XSD. cXSD describes the required data structure of input-output, inside element tag of <types>.

3) constructMessage, cMsg is constructed with the extraction of all the strings for 'requiredOperations' in between element <message…>…. </message>. Every operation has its corresponding unique Message to describe the information needed to perform the operation. Message contains one or more part that logically corresponds to the unique Message attribute inside 'Type'.

4) constructPort, cPort is constructed with the extraction of all Port strings for 'requiredOperations' in between <portType>…</portType>. cPort helps in communication of operations with the Messages. PortType contains address or connection point.

5) constructBinding, cBinding is constructed with the extraction of all the Binding string required for the operation to name its input output. cBinding is used in Type to defines the operations with its SOAP binding style (RPC/Document) and transport (SOAP Protocol).

6) constructService, cService is constructed with the extraction of Service string with the Location. cService is required for system functions that expose to the web based protocols for client to communicate with WS via Port Binding and URL.

7) constructEndDef, cEndDef is a constructed with the extraction of string at the end of definition </definitions>.

C. Combine all parts of step 'B' in sequence to form SWSDL. Algorithm to construct SWSDL is as follows.

File **ConstructSWSDL**(File *wsdl*, String[] *requiredOperations*) {
String *cStartDef, cXSD, cMsg, cPort, cBinding, cService, cEndDef*
Array of String[] *operationOFwsdl*

1 [*requiredOperations* ∈ *operationOFwsdl* | *requiredOperations* = operation for new SWSDL required to be constructed]

2 *cStartDef* = **constructStartDefinition**(*wsdl*)
cXSD = **constructXSD**(*wsdl*, *requiredOperations*)
cMsg = **constructMessage**(*wsdl*, *requiredOperations*)
cPort = **constructPort**(*wsdl*, *requiredOperations*)
cBinding = **constructBinding**(*wsdl*, *requiredOperations*)
cService = **construcService**(*wsdl*)
cEndDef = **constructEndDef**(*wsdl*)

3 *SWSDL* = *cStartDef* + *cXSD* + *cMsg* + *cPort* + *cBinding* + *cService* + *cEndDef*

return *SWSDL* }

Subset SWSDL is used for the Subset Service analysis by accessing only subset code. For example in figure 1 (c) two service are used to access Subset Service in three ways.

1) **Reduced service:** Subset of a Service can be accessed with the selected operation to access the reduce service.

2) **Combined service:** Subset of a Service composition can be accessed with the unique merge, addition or combination of operations from two or more services.

3) **Difference service:** Subset of a Service can be accessed with the difference between set of operation in two services.

Here the Service can also be api or module and operation can be method, procedure, or function.

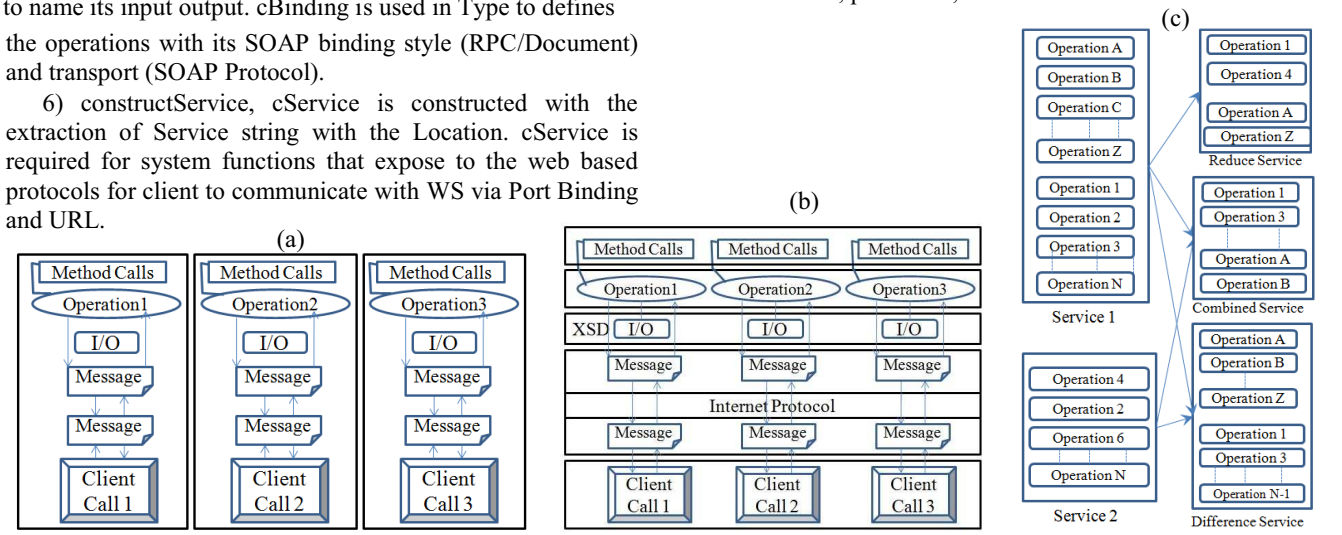

Figure 1 (a) Slicing of WS code to construct Subset Service. (b) Layers of WS to perform analysis for Subset Service. (c) Three types of Subset Service.

IV. Subset Service for Categorized Analysis

Generally, for WSA it is required to analyze its WSDL, XSD, message passing, method call graph, WS code and WS composition. Subset service is analyzed for different layers shown in figure 1 (b). Following are categorizes of WSA based on WS layers to identify all affected portions.

1) WS functional and non-functional requirement analysis: It verifies the WS and its client both separate and combined.

WS can be analyzed according to Service Level Agreement (SLA), the accessibility, audit and control, availability, load, volume and performance issues like response time, throughput, and concurrency. Analyze of code can be done for the functional and non-functional requirements with assertion to pass and fail of test case. Non-functional requirement validates scalability for the dynamic growth of users, WS compatibility and interoperability.

2) Message part WSA: Soap message posting for interaction between client and WS must be correct. Data exchange between the WS providers and its client WSDL is done according to the information provided by xml element (<message>). Communication is done with soap request and response messages according to the input-output information present in the WSDL. Input information gives parameter type and output information gives the return type of an operation.

3) Parameterized (XSD input-output) WSA: Functional analysis is performed to assert contract outputs for a particular set of inputs for a service. It can be done by manipulating XSD of WS. It is of three type input dependency, output dependency and input-output dependency.

4) Operationalized WSA: Functional requirements of WS are validated according to specification, to provide correct output. Output can be retrieved by analyzing the operation described in WSDL. Element <portType> contains one or more operation. Operation analysis is done by operation code analysis, to validate the output.

5) WS call testing: In general, WS is loosely coupled with operations but it may also be possible that two or more operations are calling each other. It might also be possible that WS is combined to another component of web world i.e. one WS is calling other WS, Website etc. Therefore, analysis of components called by a WS is also required. We need to analyze both inter-procedural called operations and methods between a WS and another WS.

V. Optimize Service Analysis using Subset service

Maintainability requires verifying the application in the production phase this can be done after the proper change impact analysis. Gather the change impact analysis to get information about inserted, deleted, or modified portions of the code. Deleted portions do not require re-executing the test cases. Inserted portions require testing new features with new test cases. Modified portions require analysis with existing, or new test cases. This information of change impact analysis will further helpful in selection of regression test cases. To use change impact analysis, we can slice WS in different loosely coupled operations with the help of SWSDL as shown in figure 1 (a). SWSDL based slicing will help in accessing Subset Service for optimization of the WS analysis process. This section described two change impact analysis based on operations and parameters of a WS.

In operationalized analysis approach, parse the WSDL and its operation code, to construct different intermediate forms of WSDL (subset WSDL). Change either in WSDL or in operation code are captured in Subset WSDLs in the form of subset operations. These operations are used for automatic selection of their respective test cases from old test suite as described in [7]. Further, these test cases combine to form the Reduce Regression Test Suite (RRTS).

All WSDL changes are captured in DifferenceWSDL (DWSDL). WS code changes are captured in UnitWSDL (UWSDL) and ParameterWSDL (PWSDL).

1. Difference WSDL is constructed from operations changed between two subsequent versions of WSDL.

{DifferenceWSDL ∈ NewWSDL | (DifferenceWSDLOperation = NewWSDLOperation – OldWSDLOperation) ∧ (same semantics i.e. port, binding, schema, message part for that operation)}

2. UnitWSDL is constructed from operations changed at statement between new and old code.

{UnitWSDL ∈ NewWSDL | (UnitWSDLOperation = changed operation at code) ∧ (same semantics)}

Operationalized analysis can be applied to any type of operation. Parameterized analysis is applied to special cases of operation where some of the parameters are interdependent on each other. Similar to the well know database technique of finding primary attribute, we can identify the special primary parameters in WS operation. These special parameters were exercised with combinations of the values for non-primary parameters. 'Primary' parameter values in WS operation can identify other parameters values uniquely. Whenever an operation has two or more non-redundant 'primary' parameters, one of them is selected as the primary parameter of that operation.

Parameterized analysis can also be useful when code is not available, for the operation whose parameter has dependencies on each other. Parameterized analysis gives mapping between WS code and WS test suite. The test suite is created according to primary parameter that will help in comprehension for user selection of test data/steps inside the test case of a test suite. Parameterized analysis is performed by identifying one of the input parameters as the 'primary' parameter. Afterwards construct test cases by varying the values of the non-primary parameters while keeping primary parameters at a fixed test value. During parameterized analysis, the identified fixed parameter was exercised with all the combinations of non-primary parameters values for regression testing of selected operations.

3. ParameterWSDL is constructed from operations changed due to change in inter-procedural flow (means change in called methods and operations).

{ParameterWSDL ∈ Input WSDL | (ParameterWSDLOperation = operations changed at data/control flows) ∧ (same semantics)}

4. ReduceWSDL (RWSDL) is constructed from user-selected operation for an input WSDL. This helps in performing regression testing and top down development.

{ReduceWSDL ∈ Input WSDL | (ReduceWSDLOperation = user selected operations) ∧ (same semantics)}

5. CombinedWSDL (CWSDL) is constructed from combination, merging or addition of unique operations in the Subset WSDLs (such as in D/U/R/P WSDL).

{CombinedWSDL ∈ Input WSDL | (CombinedWSDLOperation = unique operations merge) ∧ (same semantics)}

VI. CASE STUDY

Case study experiment designed in two parts. First, experiments for usefulness of SWDLs in WS analysis. Second, experiments conducted for Subset Service analysis.

Experiment for studying the SWSDLs. SWSDLs are used for Operationalized analysis based on the information of change impact analysis on the operations. SWSDLs are also used for Parameterized analysis of interdependencies between the parameters of an operation. SWSDLs are also used for a WS call analysis in which an operation is calling to other operation or method.

SWSDLs are properly accepted as input to the standard tools (SoapUI, JMeter, Eclipse and NetBeans). Acceptance of SWSDL as input to the standard tools validates its semantics and structure. SWSDLs are used in SoapUI and JMeter to perform regression testing. SWSDLs are also used in Eclipse and NetBeans to perform top down development. SWDSLs are used as perquisite of WS analysis for regression testing and top down development.

For regression testing based on change impact analysis we worked on following projects, namely, Eucalyptus, Amazon WS, Bible WS, currency conversion WS and Weather WS. WSDLs of different WS are given as input to the tool AWSCM. SWSDLs constructed by AWSCM is useful in two ways. First, SWSDL is usefulness when it is given as input to the SoapUI and JMeter that generates proper test suites. Further SWSDL helps in gathering of reduced test cases for the operation (changed or user selected). Second SWSDLs are useful for reduced test cases obtained in the reduced regression test suite, based on information of operation of SWSDL and the old test suite. Details are demonstrated in the author's previous paper [7].

Top down development templates from SWSDLs have been constructed for Eucalyptus, Amazon, and various other projects of different version. SWSDLs are given as input to Eclipse and NetBeans, to generate development templates, whose code required to be re-implemented. Hence, top down development can be done to the hidden WS with the help of operations in SWSDLs.

Experiment for studying the Subset Service analysis. We conducted case study for two layers, namely, operation and parameter (XSD). According to figure 1 (b), layer one and layer two require operationalized and parameterized analysis.

The operationalized analysis case study was conducted over two versions of the WS to generate the SWSDLs. The SWSDLs are used to select reduced test cases from old test suite for modified operations. Additionally, test templates (requires test data) were generated for inserted operations.

The parameterized analysis case study was conducted over WSs in table 1. For Addition of 3 digit we find all (a, b, c) are combined primary parameter. Test case for a value of primary parameter and combination values of non-primary parameters makes test suite more formal, systematic and analytic. Hence, we can properly map test data to the flow of the code.

Both approaches are prototyped as a tool, named as Automatic Web Service Change Management (AWSCM). The tool saves manual efforts by detecting the changes in the WS automatically and selecting the relevant test cases. Operationalized analysis helps to find the operations required to be retested. Parameterized analysis helps in finding out the combinations of the inputs to be exercised.

TABLE I. PARAMETER IN AN OPERATION IN BOOKSERVICE

Project	Primary parameter	Non – Primary Parameter
Currency Convertor WS	FromCurrency	ToCurrency
Global weather WS	CountryName	CityName
Bible WS	BookTitle	ChapterName and Verse
Sunset Sunrise WS	Latitude and Longitude	SunsetTime and SunRiseTime
Addition of 3 digit	a, b and c	----------

Threat to validity: We build the SWSDL for two WSDL versions (1.1 and 2.0). The subset of a WS can be more fine-grained based on analysis at the code and calling depth of functions. We discussed only two approaches based on operation and XSD layer analysis. Thus, development is required for other layers of Subset WS analysis.

VII. CONCLUSION

This paper demonstrated how to layer and slice a WS for its analysis. Five different layers described for WS analysis. Slicing of WS is done using different Subsets WSDL to access Subset Service with the selected operations. Further test cases are selected for Subset WSDL and Service. Categorized analysis reduces efforts in regression testing and top down development. In future, modules or tools can be developed with different approaches to access Subset Service. Few more SWSDLs can be created for improvement in WS development and maintenance process. Thesis [9] and tool link for detail https://sites.google.com/site/animeshchaturvedi07/research/awscm

REFERENCES

[1] B. Xiaoying, W. Dong, WT Tsai, and Y. Chen. WSDL Based Automatic Test Case Generation for Web Services Testing. Int. Workshop on Service-Oriented System Engineering (SOSE) pp. 215-220, 2005.

[2] Ruth Michael and Tu Shengru, "A Safe Regression Test Selection Technique for Web Services," 2nd Int. Conf. on Internet and Web Applications and Services on IEEE, 2007.

[3] El Bouhissi Houda and Malki Mimoun. "Reverse Engineering Existing Web service Application," 16th Working Conf. on Reverse Engineering (WCRE) pp. 279-283, Lille, Oct. 2009.

[4] R. Daniele and P. Martin, "Analyzing the Evolution of Web Services using Fine-Grained Changes," 19th IEEE Int. Conf. on Web Services pp. 392-399, ICWS, Honolulu, HI August 2012.

[5] Li, B., Sun, X., Leung, H., & Zhang, S., "A survey of code-based change impact analysis techniques," Software Testing, Verification and Reliability, vol 23(8) pp. 613-646.

[6] L. Xumin, A. Bouguettaya, X. Wu, and Li Zhou. "Ev-LCS: A System for the Evolution of Long-term Composed Services," IEEE Trans. on Services Computing, vol 6(1), 2013 pp. 102-115.

[7] Chaturvedi Animesh. "Reducing Cost in Regression Testing of Web Service," 6th CSI Int. Conf. on Soft. Engg. (CONSEG) on IEEE, Indore, Sept. 2012.

[8] Chaturvedi Animesh and Gupta Atul. "A Tool Supported Approach to Perform Efficient Regression Testing of Web Service," 7th IEEE Int. Symposium on the Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA) pp. 50-55 Eindhoven, Sept. 2013.

[9] Chaturvedi Animesh. "Change Impact Analysis Based Regression Testing of Web Services." *arXiv preprint arXiv:1408.1600* (2014).