Dynamic Web Service Slicing: a way to compute optimal Slice for Web Service

Harkishan Rathod[1], Kaushik. K. Rana[2]

Government Engineering College, Modasa, Gujarat, India
[1]hari6205@yahoo.in, [2]kaushik.rana@gecmodasa.org

Abstract **- Service Oriented Architecture is the latest technology for the today's business or enterprise. Since SOA applications are more complex, we need the way of the assuring the quality of the Web Service. Till now Service Oriented Architecture can only be tested with traditional black box testing. In this paper we are going to apply Dynamic slicing technique for white box testing of web services and computing of Dynamic slice of web service.**

Keywords **- Web Service Slicing, Slicing, Dynamic Web Service Slicing, optimal slice for web service, Java Web Service Slicer**

I.INTRODUCTION

Service Oriented Architecture are becoming increasingly more powerful and popular.[1]Service Oriented Architecture applications are growing in both numbers and complexity. Complex Web Services require more effort to ensure the quality of the service by testing the Web Service. In Service oriented Architecture the Web Service is tested in the IDE (i.e., netbeans or visual studio), although the testing facility is provided by the software it is the kind of the testing were we don't know what is happening in the testing, just we can check whether the output given by the Web Service is correct or not. This kind of testing test the Web Service until it is not so complex, but in case of complex Web Service it is necessary to know how the flow is going and which variable are being affected and by which statement, etc. White box testing in the Service Oriented Architecture may become a great need in developing and testing complex Web Service, because there is no any method for white box testing for the Web Service and the only method for testing the Web Service is black box testing. Here we are applying the program slicing technique for white box testing of the Web Service.

II.BLACK BOX TESTING IN SOA

We have already tested simple web service in netbeans IDE by right click on the web service and click on the Test service. The service now starts of being tested in following manner. Integrated Development Environment (IDE , i.e., eclips or netbeans or visual studio, or any one) create one interface for the web service i.e., a Web page that interact with the user where it asked for the input according to the input parameters of the Web Service (textbox for the input and button for the Web method, when we click on the button the web method is invoked and the result is displayed on the web page with value its data type, it also display the input values and their data type and definition of SOAP Request and SOAP Response of the Web Service). We can only check the web service only with their input values and data type and output values and their data type. No one has idea about what is going on inside, for complex problem it is impossible to find the point of the error. For such case white box testing is necessary that provide interaction with the web service and find the error in the web service.[2]

III.DYNAMIC SLICING

Dynamic slicing of any program is derived using two thins. One is slicing criterion other is input parameters. Slicing criterion consist of two <V, P>, where V is Variable used and P is the location of variable in code.[3] A static slice consist of all possible execution of program, while dynamic slice is meaningful for only a particular execution of program.[4,5]. One way to implement dynamic slicing is using program dependence graph (PDG).[6]

IV.IMPLEMENTATION

The implementation of our algorithm is divided into two stages. Stage 1 will perform lexical analysis of Web Service. And generate Tokens used in the coding of Web Service (OUTPUT.txt file). And also it generate Graph file (Web Service Dependence Graph) for the web service by parsing it. While stage 2 will compute the Dynamic slice with respect to given slicing criterion as well as input parameters.

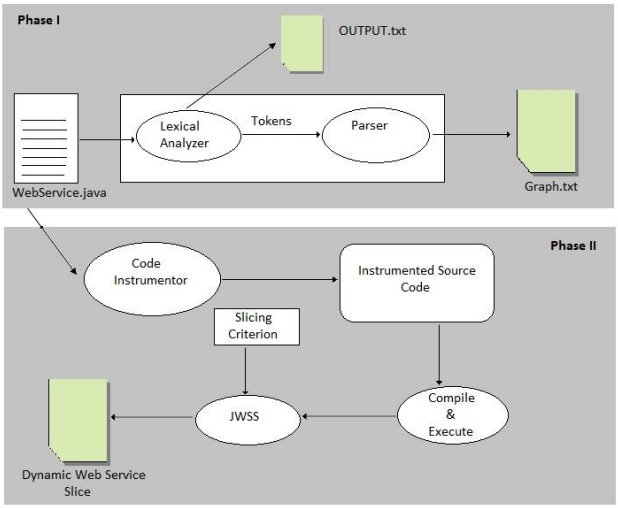

Fig. 1. Architecture of implementation

In stage 2 we have applied code instrumentation[7] for the Web Service. The code instrumentation modifies the code at runtime. We can insert our own code into the original code to track the execution of web service. Another reason for instrumentation is that, web service does not have any start point of execution, and therefore we will insert a main method that will call the web service and become the start point of execution.

In stage 2 first of all code instrumentation is applied and then we will have instrumented source code. This instrumented code then will be compiled and executed at runtime using Runtime.getRuntime().exec() method. This method is used to execute given string at runtime.

V.JAVA WEB SERVICE SLICER

The above architecture we have implemented in a single tool and we call it JWSS (Java Web Service Slicer). Since Web Services do not have user interface, we create JWSS having User Interface to interact with user.

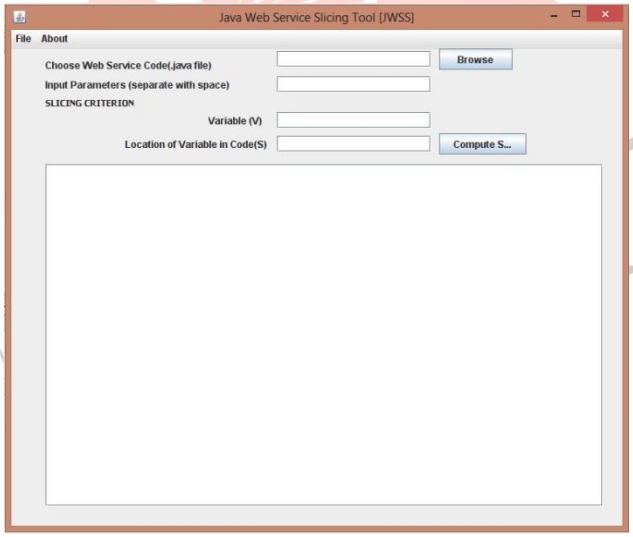

Fig. 2. Java Web Service Slicer (GUI)

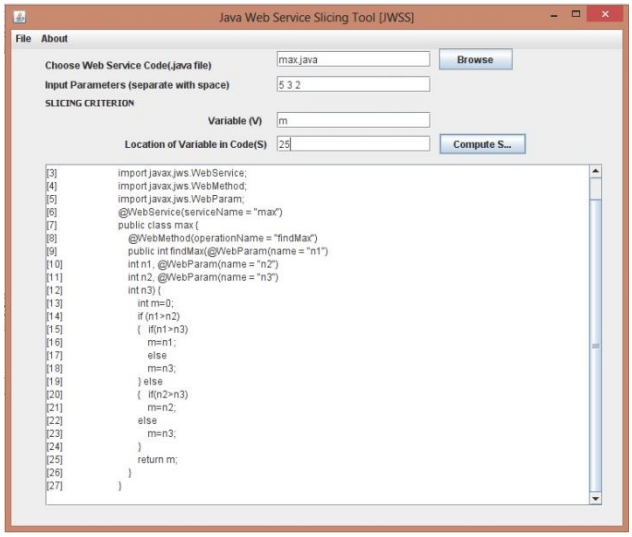

Fig. 3. Loading Web Service in JWSS tool.

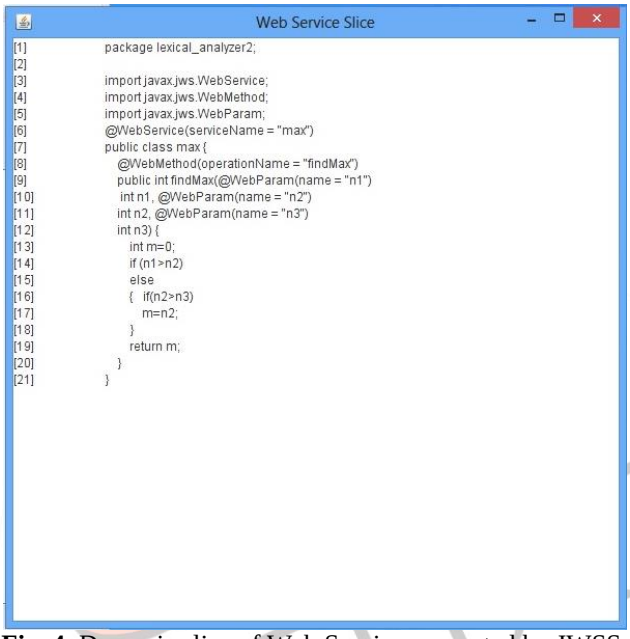

Fig. 4. Dynamic slice of Web Service computed by JWSS.

VI.RESULTS AND EVALUATION

By testing JWSS on certain web services we got following results

1) Web Service Name: hello.java

Description: Take name from user and print Hello + user name.

Table I: Result for simple hello service

Web Service	Test Case	Number of Statements	No of statement in static slice	No of statement in Dynamic slice
hello.java	"Hari"	17	17	17

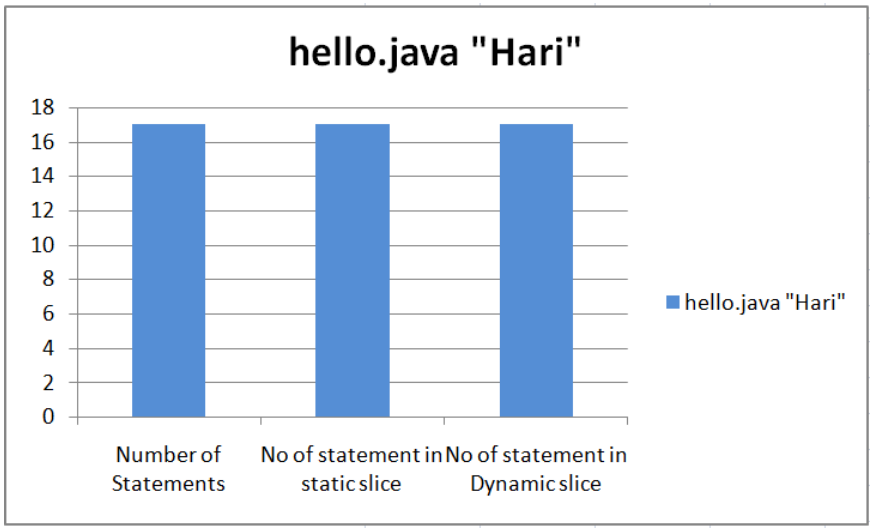

Fig. 5. Graph for Table I.

2) Web Service Name: max2.java
Description: Finds maximum number from given two numbers.

Table II: Results of max2 web service

Web Service	Test Case	Input parameters	Number of Statements	No of statement in static slice	No of statement in Dynamic slice
max2.java	t1	2,5	23	23	21
max2.java	t2	6,3	23	23	21
max2.java	t3	10,30	23	23	21

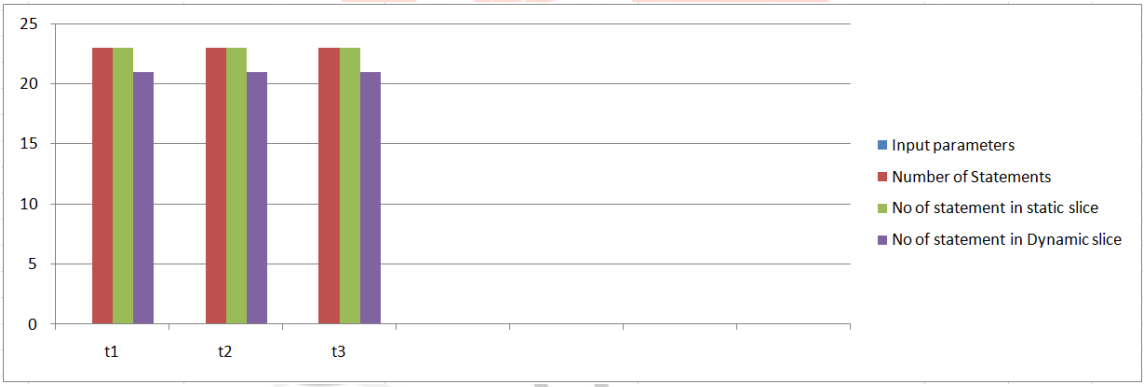

Fig. 6. Graph for Table 2.

3) Web Service Name: max.java
Description: Finds maximum number from given three numbers

Table III: Results for max Web Service.

Web Service	Test Case	Input parameters	Number of Statements	No of statement in static slice	No of statement in Dynamic slice
max.java	t1	5,3,2	27	27	20
max.java	t2	2,7,3	27	27	21
max.java	t3	3,2,4	27	27	20

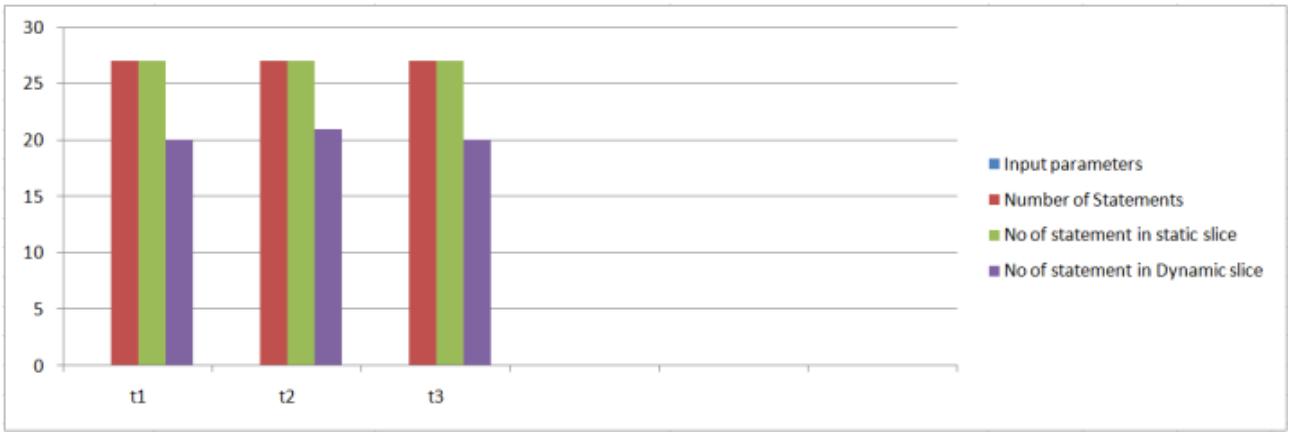

Fig. 7. Graph for Table 3

4) Web Service Name: grade.java
Description: Take marks as input and show grade based on the following rules.

Marks > 75	DISTICTION
Marks > 60 > 75	FIRST CLASS
Marks > 50 > 60	SECOND CLASS
Marks > 40 > 50	THIRD CLASS
Marks < 40	FAIL

Table IV: Result for grade Web Service.

Web Service	Test Case	Input parameters	Number of Statements	No of statement in static slice	No of statement in Dynamic slice
grade.java	t1	90	27	27	18
grade.java	t2	62	27	27	19
grade.java	t3	51	27	27	20
grade.java	t4	39	27	27	21
grade.java	t5	66	27	27	19
grade.java	t6	87	27	27	18
grade.java	t7	76	27	27	18

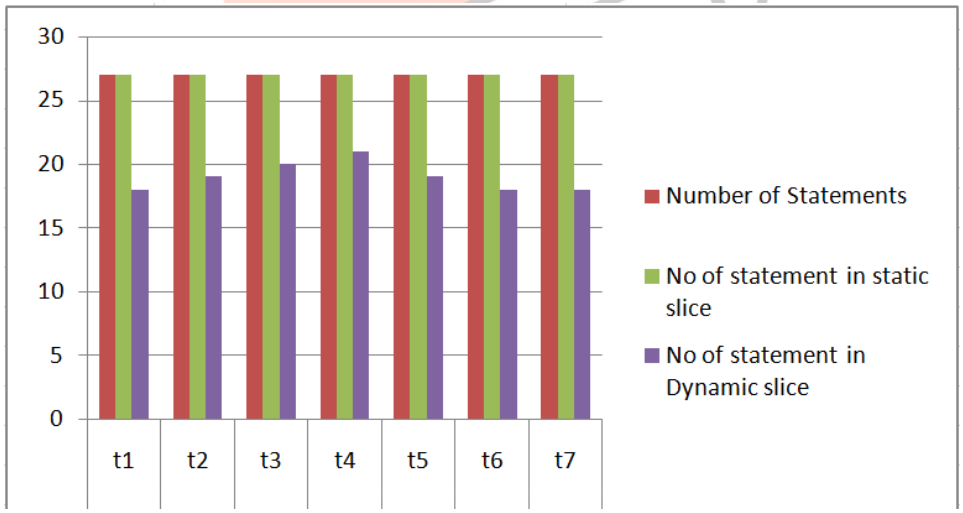

Fig. 8. Graph for Table 4.

Thus above results shows, using dynamic slice we can obtain optimum slice.

VII.FUTURE WORK

There is different kind of program slicing techniques; among them we have explored dynamic program slicing technique to compute slice of Web Service. In future we will implement dynamic program slicing technique in SOA environment. We can apply different types of program slicing techniques to test Web Service. We can also apply hybrid slicing technique that uses more than one slicing technique to compute slice of Web Service. Moreover, we can implement different approach to implement

graph for Web Service (e.g. WSDG, SCDG, and SDG). Thus we have explored Web Service quality by program slicing techniques. In future, we will implement the SDG in efficient manner to compute slice of Web Service

VIII.CONCLUSIONS

Program slicing is the technique to compute slice of the program and reduce size of the program according to slicing criterion. Applying dynamic program slicing to Web Service enables us to test Web Service internally by keeping watch on the variables used or flow of control as we do in debugging in tradition programming languages. This enables us to find errors easily in the program by applying dynamic program slicing technique to Web Service. And we can easily check errors if they exist. Thus it becomes possible to test the Web Service one by one statement. This may bring evolution in testing of Web Service.

REFERENCES

[1] Cesare Bartolini, Antonia Bertolino, Sebastiam, Eda Marchetti,"Whitening SOA Testing"
[2] Young, Michal. Software testing and analysis: process, principles, and techniques. John Wiley & Sons, 2008.
[3] Mark Weiser, "Program Slicing", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-10, NO.4, JULY 1984.
[4] D. Binkley and K.B. Gallagher. Program Slicing, Advances in Computers, volume 43. Academic Press, San Diego, CA, 1996
[5] B. Korel and J. Laski. Dynamic Program slicing. Information Processing Letters, 29(3):155-163, 1988
[6] Christian Hammer, Gregor Snelting, An Improved Slicer for Java. ACM Journal 2004
[7] David Carrera, Jordi Guitart, Jordi Torres, Eduard Ayguade and Jesus LAbarta, "An Instrumentation Tool for Threaded Java Application Server", European Center for Parallelism of Barcelona (CEPBA) 2002.