International Journal For Technological Research In Engineering
Volume 2, Issue 6, February-2015 *ISSN (Online): 2347 - 4718*

WHITE BOX TESTING OF JAVA RMI PROGRAM BY APPLYING PROGRAM SLICING TECHNIQUES

Jasmit Patel[1], Mehul Patel[2]
[1]Department of Computer Science &Engineering, Government Engineering College, Modasa
[2]Department of Information Technology, SK Patel College of Engineering, Visnagar

ABSTRACT: ***Distributed Architecture is the latest technology for the today's business or enterprise. Distributed applications are becoming the need of today's enterprise world. Success of any technology any engineering product lies on the quality of Service. Quality of Service is assured by testing. Till now in Distributed environment testing is limited by black box testing only functional requirements can be tested. If the distributed program is complex than it become difficult to test it, for black box testing does not test internal working of the program. Therefore to test complex distributed program become challenge to the tester. If we have white box testing technique in Distributed Architecture it will motivate the development of the complex program. Here we are going to apply Dynamic slicing technique for white box testing in program. These enable us to compute dynamic slice of program, Which can be useful for error finding in the code efficiently, since slice of the program is having less number of statements as compare to original code. More over Dynamic slice is based on the input parameters, and therefore it is optimal as compare to static slice.***

Index Terms: White Box Testing, Program Slicing, Program Debugging, Dynamic Slicing, Distributed Programs, Remote method invocation

I. INTRODUCTION

The complex nature of Distributed Software and speedy and rapid deployment of program inevitably leads to oversights. Testing programs minimize errors, increases the user's confidence, and hence increases quality of service. Distributed programs uses standard protocols such as Web Service Description Language(WSDL) to describe and locate service, Universal Description Discovery and Integration(UDDI) that store metadata information about Service to publish WSDL, and Simple Object Access Protocol(SOAP) to exchange messages among services[17]. Standardized protocols lead to efficient interoperability among heterogeneous systems. Testing services spans through all these layered protocols. WSDL contains abstract description (Port Type, operation, message) and concrete description(binding, port,and service). However this information are not sufficient to test distributed program, one also requires data dependency and control dependency and using these dependence we can prepare dependence graph, and applying program slicing technique.

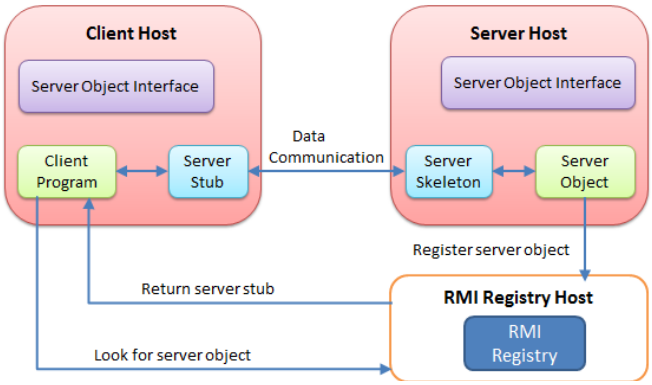

Figure-1 RMI Program Architecture [8]

Distributed applications are growing in both numbers and complexity. Complex program require more effort to ensure the quality of the service by testing. In Distributed environment the program is tested in the IDE (i.e., Net beans or visual studio or eclipse), 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 internally, just we can check whether the output given by the program is correct or not. This falls in the category of black box testing. This kind of testing test the program until it is not so complex, but in case of complex program 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 Distributed environment may become a great need in developing and testing complex programs, because there is no any method for white box testing for the Distributed program and the only method for testing the Distributed program is black box testing[5]. Here we are applying the program slicing technique for white box testing of the Distributed program.

II. PROGRAM SLICING

Set of all statements that might affect the value of a variable occurrence is totally independent of the program input values[1].A program slice consists of the parts of a program that (potentially) affect the valuescomputed at some point of interest. Such a point of interest is referred to as a slicing criterion,and is typically specified by a location in the program in combination with a subset of theprogram's variables [14].

The parts of a program that have a direct orindirect effect on the values computed at a slicing criterion C constitute the program slice withrespect to criterion C. The task of computing program slices is called program slicing.Program slicing makes various tasks like debugging, testing and

maintenance easier. Testing and debugging a smaller part of program is much easier than the complete very large program. In programs Program slicing is implemented using PDG. (Program dependence graph) The Program dependence graph is the graph whose nodes are the statement in the program and edge of the graph represents the dependence between them. The edges of the graph are of two types one is data dependence edge and other is control dependence. [2] A data dependence edge from vertex vi to vertex vj means computation is performed at vertex vi directly depends on the value computed at vertex vj. In other words the computation at vertex vi uses a variable, that is defined at vertex vj, and also there is an execution path from vj to vi. A control dependence edge from vi to vj means that node vi may or may not be executing depending on the Boolean outcome of the predicate expression at node vj.

```
        begin
S1:         read(X);
S2:         if (X < 0)
            then
S3:             Y := f1(X);
S4:             Z := g1(X);
            else
S5:             if (X = 0)
                then
S6:                 Y := f2(X);
S7:                 Z := g2(X);
                else
S8:                 Y := f3(X);
S9:                 Z := g3(X);
                end_if;
            end_if;
S10:        write(Y);
S11:        write(Z);
        end.
```

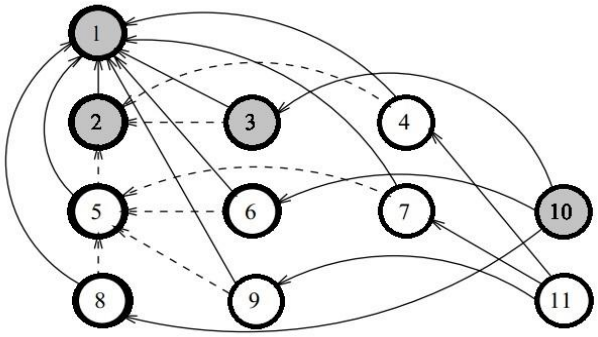

Fig. 2. Program Dependence graph for Figure 1 for input value X=5, where slice consists of statements 1, 2, 3 and 10. Here solid edge represents data dependence and dashed edge represents control dependence. And darken node represents the slice of the program. [1]

A. Static Slicing vs Dynamic Slicing.

It has been shown that dynamic slicing is efficient than static slicing as it takes less memory because the size of the slice produce dynamically is small. Finding all statements that really affected the value of a variable occurrence for the given program inputs[3]. In dynamic slicing always less or equal number of statements than static slicing.

Technique	Pros	Cons	Applications
Static Program Slicing	Easy to Understand	Static Slice is not minimal slice	Software Quality Assurance, Software maintenance, Program Integration
	Easy to Implement		
	Slice is computed without making assumption regarding program's input		

Dynamic Program Slicing	Slice is computed with making assumption regarding program's input	Little difficult to implement than Static Slicing	Testing, Debugging, Performance Improvement
	Dynamic Slice is minimal slice		

III. WHITE BOX TESTING OF RMI(REMOTE METHOD INVOCATION) PROGRAMS USING DYNAMIC SLICING

We can test the one of the distributed program is Remote Method Invocation. In this computing two different programs used are server side program and client side program. Server is register at the registry and client lookup the object of the server. After fetching the object from registry, client can invoke the method of server with the use of intranet or internet. Since in Server side program code is directly not available at the run time, therefore we have to make some assumption as below.

- The server code is available to us and we are testing RMI program before using it.
- The computer on which the Server code is published (RMI directory service – Registry) is available and it is working correctly.
- Our method of white box testing in Distribution environment is the testing before we deploy the program.

We have introduced three new graphs to understand the dependency of each other and developer can find the issue easily and solve error immediately.

A. Server Dependence Graph

In order to make slice of a RMI program we would have to use deferent type of directed graph, Server Dependence Graph. (SDG). In SDG following types of dependencies are identified

- Data Dependence
- Flow dependence
- Callee Dependence

Data Dependence is a type of dependency in which particular statement depends on any data or variable. Flow dependence is a type of dependency in which particular statement depends on the flow of the execution. Here we go for the testing of the server program before connecting and therefore we don't have to access to the service code of the remote machine. In SDG the nodes are the statement number, and edges are of three types. One is data dependent edge, second is flow dependent edge, and the third is callee dependent edge. For Data dependence if there is a node from vj to vi then we can say that statement vj depends on the variable from statement vi. For if flow dependence, if there is a node from vj to vi then we can say that statement vj depends on the flow of execution from statement vi or in other words statement vj depends on the result of the local called server program from statement vi. For callee dependence, if there is a node from vj to vi then we can say that statement vjn depends on the result of server program called from remote machine from statement vi.

B. Client Dependence Graph

There is other kind of the directed graph at client called Client Dependence Graph. The control/flow moves from Client Dependence Graph to server Dependence Graph when the web method is called. There are only two kind of the dependence in the CDG. One is Data dependence and other is Control dependence. Data Dependence is represented as solid line. Control dependence is represented as dashed line. Control dependence occurs in the statements of looping or branching or jumping or when server program is called.

- Data Dependence
- Control dependence

C. Distributed Program Dependence Graph

The Distributed Program Dependence Graph contains Server Dependence Graph (SDG) and Client Dependence Graph (CDG). SDG represents the whole structure of the RMI server program call and CDG represents the whole structure of the RMI Client program call. The Registry call is added in new Distributed Program Dependence Graph. Server can add new service in Registry (Directory service).

- Callee dependence
- Registry dependence

IV. ALGORITHM FOR DISTRIBUTED DYNAMICSLICING (DDS)

Input: Server Code, Server Implementation Code, Client code, Slicing Criterion (< S, V >)
Output: Dynamic slice
Stage 1: Construction of DPDG.
Stage 2: Compute Slice for DDS

Stage 1: Construction of DPDG
a) Construct SDG
1. Node construction
    a. Create two special nodes start stop
    b. For each statements of RMI Program do
        i. Create a node s
        ii. Initialize the node with its type, list of variables used or defined, and its scope
2. Add data dependence or flow dependence or registry dependence
    For each node ui do following
    For each node uj do following
        a. Add data dependence edge (ui, uj), if one or more variables or data is used at nodeuj from node ui

b. Add flow dependence edge (ui, uj), if flow moves from node ui to node uj

c. Add Registry dependence edge(ui, registry), if node ui can bind the object with registry.

b) Construct CDG

1. Node construction

a. Create two special nodes start and stop

b. For each statement s of RMI program, do

i. Create a node s

ii. Initialize the node with its type, list of variables used or defined, and its scope

2. Add data dependence or control dependence or callee dependence

For each node ui, do following

For each node uj, do following

a) Add data dependence edge (ui, uj), if one or more variables or data is used at node uj from node ui.

b) Add control dependence edge (ui, uj), if control moves from node ui to node uj.

c) Construct DPDG

1. In CDG if program is called from node ui to node vi then add callee dependence edge (ui, vi)

2. If program returns data from node ui to client node vi add callee dependence edge (ui, vi)

Stage 2: Compute Slice for DDS.

1) For every variable v used at node uj from ui do

a. Mark data dependence (ui, uj), if any data or variable used at node uj from node ui.

b. Mark control dependence (ui, uj), if flow of control depends on node ui from node uj.

c. Mark callee dependence (ui, vi), if program is called from node ui node vi.

2) Trace execution.

3) Remove unmarked node from distributed program.

4) Set of Marked nodes is dynamic slice.

V. CHALLENGES TO WHITEBOX TESTINGOF DISTRIBUTED PROGRAMS

A. Unavailability of the server code:

In Distributed Architecture program is described by the Web Service Description Language (WSDL) and located using Registry (i.e. UDDI). In such environment only server interface is available, Server code is not available to the third party (i.e. Service Consumer) in that case we have to take assumption about the availability of the server code at the time of testing[5].

B. Reliability of the Registry:

Registry is the service provider so that it can working as per the server bind the object with registry and client lookup the server object from the registry.

C. Knowledge of Distributed programs:

As we have going to test with program slicing is the white Box testing techniques so that it may require the programming knowledge of the working of distributed program.

D. Composition of the server code:

1) Intra domain: In Intra domain Server, Server is on the same domain, and can be used for that domain only.

2) Inter domain but intra network: In inter domain but intra network server is deployed on another machine on the same network. This server can be access with the speed of the network speed. It is also affected by the reliability of the network.

3) Inter domain but internetwork: In inter domain and network the server is deployed on another machine in the different network, in such case it is highly dependence on speed of network and reliability of the network.

VI. CONCLUSION & FUTURE WORK

This method of slicing allows us to compute slice of the RMI Program. By computing slice of the RMI program we can test the program by white box testing, thus it becomes possible to test the program with one by one statement. And we can keep watch on any variable or the flow of the control as we do in debugging in traditional program (i.e. C/C++). We can also improve performance of the program by removing unwanted code. Using the dynamic slicing of white Box testing find the fault or bug easily and solve immediately. In future, we are going to implement white box testing for RMI distributed program. We need to make the tree representation. We need to prepare three graphs, Server Dependence Graph (SDG), Client Dependence Graph (CDG) and Distributed program Dependence Graph (DPDG). Moreover since We would have to prepare user interface of program according to its input parameters and output parameters along with their data types.

This type of technique can be applied to find the slice of the RMI program. Here we apply dynamic slicing technique; we can also apply other techniques to compute the slice for the distribution program. Development of this technique will lead to the evolution in the field of Distributed Architecture. There are different ways to implement the graph in computer world we can apply the different techniques to implement the graph, and we can use the most suitable technique.

REFERENCES

[1] Hiralal Agrawal, Joseph R. Horgan, “Dynamic Program Slicing”, ACM SIG-PLAN'90 Conference on Programming Language Design and Implementation, New York, June 20-22 ,1990.

[2] Mark Weiser, “Program Slicing”, IEEETRANSACTIONSONSOFTWAREENGINEERING,VOL.SE-10, NO.4, JULY 1984.

[3] Vijay K. Garg ,Neeraj Mittal, “On Slicing a Distributed Computation”, 21st International Conference of distributed computing systems, April 2001.

[4] Harkishan Rathod, "Testing Web Services by

Applying Program Slicing", International Journal of Advance Research in Computer Science and Management Studies, Volume 2, Issue 1, January 2014

[5] Durga Prasad Mohapatra, Rajib Mall, Rajiv Kumar, "Distributed Dynamic Slicing of Java Programs", International Conference, ICDCIT 2004

[6] Mrs. Sonam Jain, Mr. Sandeep Poonia, "A New approach of program slicing:Mixed S-D (static & dynamic) slicing." International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 5, May 2013

[7] Dr. Roger Eggen, Dr. Maurice Eggen, "Efficiency of Distributed Parallel Processing using Java RMI, Sockets, and CORBA", International Conference on Parallel and Distributed Processing Techniques and Applications 2001.

[8] S.S. Barpanda D.P. Mohapatra. "Dynamic slicing of distributed object-oriented programs." IET Software, April 2011.

[9] Sanjay P. Ahuja , Renato Quintao. " Performance Evaluation of Java RMI: A Distributed Object Architecture for Internet Based Applications." 0-7695-0728-WOO$ 10.00 0 2000 IEEE.

[10] Mohapatra, Durga Prasad, Rajib Mall, and Rajeev Kumar. "An overview of slicing techniques for object-oriented programs." Informatica (Slovenia) 30.2 (2006): 253-277.

[11] Chengying Mao. "Web Service-based Software." 978-1-4244-5299-6/09/$26.00 (c)2009 IEEE.

[12] Neeraj Mittal, Vijay K. Garg. "Software Fault Tolerance of Distributed Programs Using Computation Slicing." 23rd International Conference on Distributed Computing Systems (ICDCS'2003).

[13] Frank Tip, "A Survey of Program Slicing Techniques".

[14] Durga Prasad Mohapatra, Rajib Mall and Rajeev Kumar. "A Novel Approach for Computing Dynamic Slices of Object-Oriented Programs with Conditional Statements." IEEE INDIA ANNUAL CONFERENCE 2004

[15] Riazur Raheman, Amiya Kumar Rath, M Hima Bindu. "An Overview of Program Slicing and its Different Approaches.", International Journal of Advanced Research in Computer Science and Software Engineering , Volume 3, Issue 11, November 2013.

