International Journal of Combined Research & Development (IJCRD)
eISSN:2321-225X;pISSN:2321-2241 Volume: 4; Issue: 5; May -2015

# ABSTRACTION OF TEST CASES FROM INPUT JAVA PROGRAM

**Dr. R N Kulkarni, Nidhi Jain C, Rashmi G, Vaishali B J, Zakiya Niyazi**
Department of Computer Science & Engineering

BITM, Ballari, Karnataka, India

**Abstract:** Now a day, we find there is a tremendous change in the software industry, where both hardware and software are changing rapidly. The organizations which are already automated have to make changes in their processing activities. From the last two decades a new discipline has been started where the developing organization will develop the required application for the client, subsequently they hand over the testing part to the third party organization. The testing organization consists of the skilled and experienced peoples whose responsibility is to test the software against the requirement and after testing the test reports will be sent to the developing organization. While preparing the test cases, the testing organization spends a lot of time in understanding application code or program code; design the test cases and finally testing the application. To overcome this problem, in our project we are proposing an automatic tool which abstracts the test cases from the input java program. This abstraction is carried out by restructuring input program followed by the abstraction of test cases and retesting these abstracted test cases on the program to verify for its correctness.

**Key Words:** Restructuring, Abstraction, Test cases, Correctness.

## 1. INTRODUCTION

The traditional approach of the software testing comprises of five stages – planning and control, test analysis and design, test implementation and execution, evaluation and reporting and test closure. In the software testing one of the challenging tasks is to select the test inputs which need to be pre-defined. If one of the test cases does not hold good then entire testing processes need to be executed for the test suite which increase the complexity to analyze the code and also help to analyze the efficiency of test-cases. To overcome these drawbacks of traditional approach a tool is generated to analyze the program written in Java-language, which abstracts test case by restructuring the program without affecting the functionalities, evaluates the correctness of the test cases abstracted and maximizes the verification of code with minimum complexity.

The purpose of a test case is to document the steps and conditions under which a particular test scenario must be executed, along with the expected result. Test cases are written in the test analysis and design phase of the software testing cycle, wherein the test objectives are transformed into test conditions. Effective test cases are necessary to achieve complete quality verification and validation of the application under test.

Abstraction of test cases also provides the users to easily reproduce the steps that were undertaken to uncover a defect that as detected during test. Test-case is basically a description of test where it has the components that describe the input, event or action and expected response to determine if feature is working correctly.

## 2. LITERATURE SURVEY:

In the paper [1], the author discusses about various approaches for the generation of test cases viz, UML based, graph based, formal methods, web application, web service, and combined. From this paper we are using web based approach to abstract the test cases for our project. But this paper doesn't discuss about the approach to abstract the test cases from specific program.

In the paper [2], the author discusses about various approaches for the generation of test cases from UML diagram viz, UML Structured Diagram, UML Behavioral Diagram and Combinational Approach. But using UML diagrams for generation of test cases will not attain maximum test coverage and even if it generates significant number of tests it will have less test coverage.

In the paper [3], the author presents the implementation techniques of a tool called 1-click test executor, which fully automates all processes necessary for Java program testing. The tool significantly reduces the time required for high-coverage testing, from several minutes to just a few seconds. But this paper doesn't discuss about checking the correctness of the test cases.

In the paper [4], the author presents comprehensive techniques such as the mutation testing, input/output analysis and data mining for evaluation of test cases that are generated. From this paper we are using a technique called the input/output analysis for evaluation of test cases.

In the book [5], the author has given an abstract model of the 'traditional' testing process which consist of four steps as a) Design Test Cases, b) Prepare Test Data, c) Run program with test data and d) Compare results to test cases. We have used this process for abstraction of test cases from input java program.

In the book [6], the author discusses about various testing methodologies, levels of testing and how test case information should be displayed.

## 3. TERMINOLOGY

a. **Restructuring:**
Restructuring is the transformation of one representation form to another at the same abstraction level.[7] The input java code is restructured by removing the comment lines, spaces, and it breaks the line such that only one statement is included in each line. The transformation preserves the functionality of the program. At the end of this phase the input java program is in structured from.

b. **Abstraction:**
Abstraction is the act of representing essential features without including the background details or explanations. Test cases are abstracted from the structured code and the output is displayed in the form of table, consisting of test id, possible input, expected output and remarks.

c. **Retesting:**
Test cases that are abstracted are re-tested, by giving the possible input from the abstracted table to the program. The output obtained from the program is compared for the correctness with the output of the abstracted table.

d. **Test suites:**
It is the collection of the test cases that are intended to be used to test a software program to show that it has some specified set of behaviour.

## 4. PROPOSED METHODOLOGY:

Our project proposes a methodology that abstract test cases from the input java program. The tool accepts the input program and automatically abstracts the test cases without any user intervention in the process and checks for the correctness of the test cases obtained. The methodology comprises the following steps:

**4.1. Restructuring:** Initially scan the program, restructure the code by placing the methods in call in sequences order and arranging the classes in inheritance sequence.

/* Algorithm for the Restructuring */
Input: Executable 'java' program
Output: Restructured java code stored in a file

1. [Restructuring of the java code]
1.1 Remove the comment lines, if any.
1.2 Remove the blank lines, if any.
1.3 Break the line, if multiple statements are present in a single line or if selection statements (if, while, do-while, for) are encountered.
1.4 Assign line number to each physical statement of the program.
1.5 Store the obtained output in a file.

**4.2. Abstraction:** Abstraction is the act of representing essential features without including the background details or explanations.

/* Algorithm for the Abstraction */
Input: Restructured file
Output: Test suite in a XL file

1. Scan the entire program and read each statement.
1.1 Identify the variables from the restructured program.
1.2 Based on the data type of variable, boundary value analysis is done within the range of that type.
1.3 For the identified variables in a program test suite is generated and it sent to XL file.

**4.3. Retesting:** Test cases that are abstracted are re-tested, by giving the possible input from the abstracted table to the program. The output obtained from the program is compared for the correctness with the output of the abstracted table.

/* Algorithm for the Retesting */
Input: One entry from an abstracted test cases table
Output: Correctness of test cases in XL-sheet
1. [Retesting of abstracted test cases]
1.1 One of the possible outcome of the abstracted test case table is taken.
1.2 The above outcome is given as an input to the executable java progam.
1.3 Obtained output is cross verified with the output of the abstracted test case table.

## 5. CASE STUDY

The proposed procedure is implemented for number of 'java' programs and the results we got are correct and complete.

**[A sample 'java' program]**

```
class Addition
{
   @Test
    public static void main(String[ ] args)
    {
          int var1 = 10;
          int var2 = 20;
          int result = var1 + var2;
           System.out.println("A ="+var1+
           "B ="+var2+     "C ="+result);
     }
}
```

**OUTPUT**


Fig 5.1 Home page

Fig 5.2 Restructuring

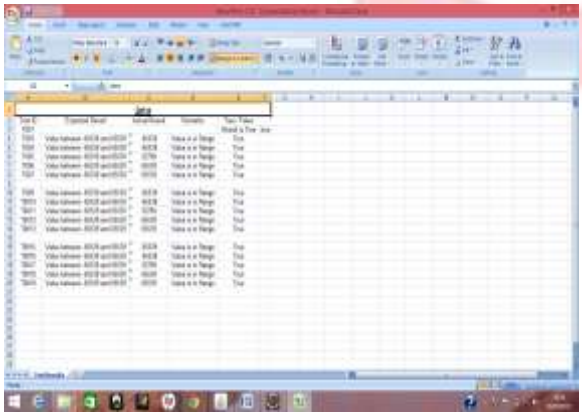

Fig 5.3 Abstraction of test cases

## 5. CONCLUSION

In this paper we have proposed an automatic tool for the abstraction of the test cases from the input java program. This abstraction is achieved by restructuring, then abstraction of test cases and re-testing the program using the abstracted test cases. The tool is tested for its correctness and completeness.

## REFERENCES:

[1] Ebrahim Shamsoddin-Motlagh *"A Review of Automatic Test Cases Generation"* , International Journal of Computer Applications (0975 – 8887) Volume 57– No.13, November 2012.

[2] S. Shanmuga Priya, Dr. P. D. Sheba, "*TEST CASE GENERATION FROM UML MODELS – A SURVEY"* International Journal of Emerging Technology and Advanced Engineering Website: www.ijetae.com (ISSN 2250-2459 (Online), An ISO 9001:2008 Certified Journal, Volume 3, Special Issue 1, January 2013)

[3] Supasit Monpratarnchai, Shoichiro Fujiwara, Asako Katayama, Tadahiro Uehara, *"An Automated Testing Tool for Java Application Using Symbolic Execution based Test Case Generation"* 2013 20th Asia-Pacific Software Engineering Conference

[4] Mohammad Reza Keyvanpour, Hajar Homayouni and Hossein Shirazee, *"Automatic Software Test Case Generation: An Analytical Classification Framework"* International Journal of Software Engineering and Its Applications, Vol. 6, No. 4, October, 2012.

[5] Ian Sommervillie,"Software Engineering", 9th Edition, Addison-Wesley.

[6] Paul C.Jorgensen, "Software Testing- A Craftsman's approach", third edition, Auerbach publications.

[7] Dr. Shivanand M. Handigund and Rajkumar N. Kulkarni, ] "An Ameliorated Methodology for the Abstraction and Minimization of Functional Dependencies of legacy 'C' Program Elements" International Journal of Computer Applications (0975 – 8887) Volume 16– No.3, February 2011

.