International Journal of Innovative Technology and Exploring Engineering (IJITEE)
ISSN: 2278-3075 (Online), Volume-9 Issue-5, March 2020

Analysis of Statements of Software Based on the Prioritization Strategy of Test Cases

Sasmita Padhy, Satya Narayan Tripathy, Sisira Kumar Kapat, Susanta Kumar Das

Abstract- Software testing is a major process in every software development cycle, which is a very expensive procedure. Throughout this phase software testers as well as development companies are eager to verify the software from various viewpoints for repeated testing. But detailed testing requires code execution along with all valid inputs in the form of test cases which is not possible because of resource limitations. The test cases are automatically generated in many of the software. One of the important concerns for any coder during the maintenance phase is the selection of successful test cases for code execution. This paper includes an algorithm that is designed to figure out the statement coverage of a code with proper inputs by executing the priority based approach which involves sorting the inputs in ascending order so as to have an appropriate method to test the code for maximum statement coverage.

Keywords: statement coverage, Software Testing, Test case prioritization

I. INTRODUCTION

One of the major concerns of software testing is to achieve maximum statement coverage speedily. Since the problem is complicated because testers need to determine the input data cleverly, that can execute the processes faster towards different portions of applied code which contains more statements [1]. Detailed testing involves execution of code along with all possible selection of values for code variables which is not possible because of resource limitations. In most applications the test cases are generated automatically. Also the selection of appropriate successful test cases to validate the code is a major cause of concern in the maintenance phase [2]. White box testing is a process which requires the compilation of all the statements minimum once in the source code. Statement coverage is a type of white box testing which is a measure to calculate and forecast the number of statements that are compiled in the source code. With this technique one can examine what the source code is estimated to do and what not. This technique can also be used to prove the quality of code and stream of different paths in the code [3]. Statement coverage will not be calculated in a code which includes missing statements, dead code and unused statements [4].

Statement coverage implies that the number of test cases that have compiled in the statements. Statement coverage is not equals to 100% implies that, all lines of code have not been compiled properly [5]. So this can be achieved by recognizing every line of code in a cyclic manner and compiling the minimum number of set of test values. Also it can be calculated by using the formula,

Statement coverage in percentage = 100 * (number of compliable statement compiled/ total number of compliable statements) [6].

One of the major benefits of measure of statement coverage is its ability to immensely separate a section of code that could not be compiled otherwise. Also statement coverage compiles only conditions that satisfy the statements. We can also recognize some of the statements compiled and where the code cannot be compiled because of hurdles or obstacles that are there in the program [7].

Software testing generally aims at two important criteria that are to verify the application is developed as per the customer requirements and to identify the bugs. So this can be only ensured through “software testing”. But testing is a very long process, requires a lot of infrastructure that can be costly and also tedious job. So it should be chosen based on likelihood or danger of the application. For this white box testing is normally used since it is capable of validating structure, decision expectations and finding bugs and errors and operational errors in the software. This technique generally includes analysis of interior structure and architecture of section of software. So that poor key management type application errors can be exposed. It also operated based on the implementation of the application. White Box testing has a lot of advantages such as, error detection in hidden codes, deleting unnecessary lines of codes, etc. which increases the complexity [8].

A. Presenting Data for Testing:

Complete coverage of path seems to be done easy in exemplary world but in real cases complete testing of applications covering all paths seems tougher .The main reason for this is the requirements of having test data that is necessary to implement a feasible path by virtue of interactions amongst all decisions across a procedure. So this method is not as simple as it seems since introduction of test data which would further give a definite path in a procedure needs attention [9].

Revised Manuscript Received on March 30, 2020.
* Correspondence Author

Dr. Sasmita Padhy, Associate professor, Department of CSE, Vignan Institute of Technology and Management, Berhampur, Ganjam, Odisha. e-mail: pinky.sasmita@gmail.com
Dr. Satya Narayan Tripathy (0000-0002-6005-687X**)**, Assistant Proffessor, Department of Computer Science, Berhampur University, Odisha. e-mail: snt.cs@buodisha.ac.in
Er. Sisira Kumar Kapat (0000-0002-2489-6214**)**, Lecturer, Department of CSE/IT, Uma Charan Patnaik Engineering School, Berhampur, Odisha, e-mail: skk.rs.cs@buodisha.edu.in
Dr. Susanta Kumar Das , Reader, Department of Berhampur University, Odisha, email: skd.cs@buodisha.edu.in

© The Authors. Published by Blue Eyes Intelligence Engineering and Sciences Publication (BEIESP). This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

The following criteria are followed by the coders to simplify testing process [9]:

- To make the code simple and understandable for testers.
- Not to have cyclic loops of more than ten.
- There should not be many complicated paths
- Path minimization is required
- Data redundancy must be avoided
- Decisions must not overlap with each other.

II. LITERATURE SURVEY

Black box, white box and gray box are some of the frequently used software testing techniques to test different type software.

White Box Testing tests the functioning of the software as well as its interior architecture. If the code is implemented, it can find the possible errors; but if the code is not implemented properly it can't find the errors, which is a limitation of white box testing. The working of the internal code of the software is done by another technique known as black box testing. This can be done throughout all the levels of the software. Gray box technique is simply the union of white and black box testing.

One of the main purposes of statement coverage is that it verifies whether all the sections of the codes are executed properly and ensures that which section of the code failed to execute in the program. Again here one of the major drawbacks of this is that it can't verify all the conditions in a single block

III. RELATED WORK

Muhammad Shahid and Suhaimi Ibrahim gives attention to regression testing as well as prioritize regression techniques and tools for measurement of coverage and they have given an advanced algorithm to assign priority to the test case which relies on coverage operations so as to enhance regression tests used in maintenance phase. They imply that if the test values cover more methods then it has more probability to detect errors than earlier [11]. Sudesh Kumar, Nupur Gupta presented a study of black box testing method to bring about the test values and statement coverage principle to minimize test cases that will cut cost as well as time that is spent on testing [12]. Statement coverage implies that the calculation of percentage of statements that are executed by test inputs. Statement coverage is not equals to 100% implies that all the lines of code have not been compiled [5]. Estimated benefits of statement coverage are it is capable to segregate the section of code that could not be executed [8] [13].

IV. PROPOSED ALGORITHM

Here we have presented an algorithm to find out the statement coverage of a code and evolve a graphical viewpoint by considering various test cases and sorting them in increasing order to identify the maximum coverage of the code. The presented algorithm has been described below. The presented algorithm is applied in various programs on object oriented languages so as to find out the maximum statement coverage on a certain test input criteria by implementing more and more test inputs one can find out proper test cases. Consequently one will be able to amend the instructions so as to achieve the required code.

The proposed algorithm is as shown below:

Input	:	Initial Test suite TS, number of executable statement covered by a test case, total number of executable statement.
Output	:	Prioritized Test suite TS`

```
step1  :  start
step2  :  define TS` = empty
step3  :  for each test case ts € TS do
step4  :  sum of statements covered
step5  :  calculate statements coverage percentage = (Number
          of executable statement executed / total number of
          executable statements) * 100
step6  :  end for
step7  :  arrange TS in increasing order positioned on the
          percentage of statement covered by all the test case
step8  :  let TS` = TS
step9  :  end
```

The program to find the largest among three numbers using Java is presented in figure-1.

```
1.    import java.util.Scanner;
2.    class LargestOfThreeNumbers{
3.    public static void main(String args[]){
4.    int num1,num2,num3;
5.    System.out.println("Enter three integers ");
6.    Scanner in = new Scanner(System.in);
7.    num1 = in.nextInt();
8.    num2 = in.nextInt();
9.    num3 = in.nextInt();
10.   if ( num1 > num2 && num1 > num3 )
11.   System.out.println(" num1 is largest.");
12.   else if ( num2 > num1 && num2 > num3 )
13.   System.out.println(" num2 is largest.");
14.   else if ( num3 > num1 && num3 > num2 )
15.   System.out.println("num3 is largest.");
16.   else
17.   System.out.println("Entered numbers are not
  distinct."); }    }
```

Fig. 1.Program to find largest among three numbers using Java

Here the 3 input values for the above program is taken in line no 5, 7 and 8 as num1, num2 and num3 respectively. So the 3 input variables to form test cases can be arranged in 3! Ways which is 6.

So the possible test cases are:

TC1 = 5 7 8
TC2 = 7 5 8
TC3 = 8 5 7
TC4 = 7 8 5
TC5 = 8 5 7
TC6 = 8 7 5

TC7 is taken as special case for non distinct value of variables like (8 8 7).

In this paper we have explained the detailed implementation of all the test cases of the above program on the basis of proposed algorithm for finding out the statement coverage of the code.

Table-1: Traceability between Test Case and Test Statements

TS/TC	TC1	TC2	TC3	TC4	TC5	TC6	TC7
TS1	1	1	1	1	1	1	1
TS2	1	1	1	1	1	1	1
TS3	1	1	1	1	1	1	1
TS4	1	1	1	1	1	1	1
TS5	1	1	1	1	1	1	1
TS6	1	1	1	1	1	1	1
TS7	1	1	1	1	1	1	1
TS8	1	1	1	1	1	1	1
TS9	1	1	1	1	1	1	1
TS10	1	1	1	1	1	1	1
TS11	0	0	1	0	1	1	0
TS12	1	1	0	1	0	0	1
TS13	0	0	0	1	0	0	0
TS14	1	1	0	0	0	0	1
TS15	1	1	0	0	0	0	0
TS16	0	0	0	0	0	0	1
TS17	0	0	0	0	0	0	1

Note: TS: Test Statements, TC: Test Case.

The following results can be derived from the above example:

Total number of statements in the program: 17
Total number of test cases for the program: 07

Statement covered by each test case:

TC1=13 TC2=13 TC3=11 TC4=12
TC5=11 TC6=11 TC7=14

Calculating the statement coverage in percentage by the following formula:

Statement coverage percentage = (Number of compliable statement compiled / total number of compliable statements) * 100.

TC1= 76.47% TC2= 76.47%
TC3= 64.70% TC4= 70.58%
TC5= 64.70% TC6= 64.70%
TC7= 82.35%

We can infer from the above example that test case 7 covers 82.35% statement in the program. So it can be inferred that it is an appropriate test input for the software tester to attain desired results in form of Test suite. If merged together, some test cases form test suite like TC1 or TC2, TC3 or TC5 or TC6, TC4 and TC7 together then there will be 100% statement coverage for the above proposed code.

Presenting the above outcomes in Bar-graph in figure-2:

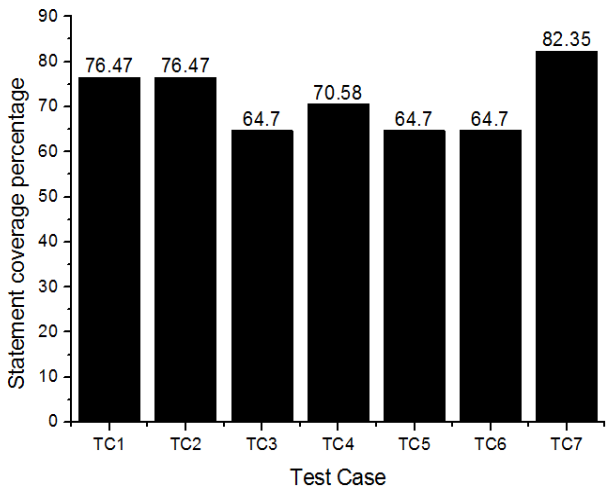

Fig. 2.

Showcasing the same result in form of Bar-graph in below figure 3 by prioritizing them in ascending order:

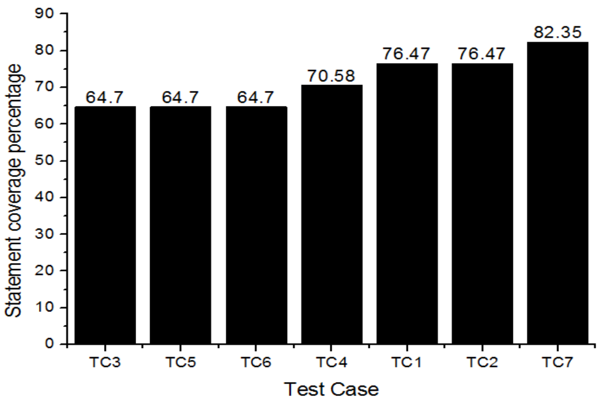

Fig. 3.

Here we conclude that prioritize test case cover more statement coverage in increasing order. Consequently we can find more similar or appropriate test case (TC7 in our example) which has maximum coverage in the program and present the Test Suite to test the software, in a good way and to distinguish the unwanted code from the program to have an enhanced and errorless code for ever.

VI CONCLUSION

In this paper, we have proposed an appropriate technique to find out percentages of statement covered in each test case and implementing the priority based technique to sort them in increasing order so as to find out a definite approach to test the program for achieving maximum statement coverage.

Despite the fact that this is an initial and simple approach to find out the statement coverage we can further upgrade this approach by working on more such statement coverage based on priority based technique which will immensely help testers to sort out the percentage of unwanted statements that are there in the program which in turn can be immensely helpful for the development organization to make software of customer requirements.

REFERENCES

1. Park, Sangmin, et al. "CarFast: achieving higher statement coverage faster." Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. ACM, 2012.
2. S. u. R. Khan, A. Nadeem and A. Awais, "TestFilter: A Statement-Coverage Based Test Case Reduction Technique," 2006 IEEE International Multitopic Conference, Islamabad, 2006, pp. 275-280.doi: 10.1109/INMIC.2006.358177
3. http://www.zyxware.com/articles/4161/what-is-statement-coverage-in-testing (9-sept-2016/10:30 AM)
4. http://sqa.fyicenter.com/FAQ/ISTQB-Software-Testing-Exam/Statement_ Coverage_will_not_check_for_the_follow.html (9-sept-2016/11:30 AM)
5. Control Flow Testing available at: http://www.computingstudents.com/notes/software_analysis/control_flow_testing.php
6. https://www.scribd.com/doc/27434880/Statement-Coverage-Branch-Coverage-Condition-coverage-tutorials-for-Software-testers (9-sept-2016/10:30 AM)
7. http://istqbexamcertification.com/what-is-statement-coverage-advantages-and-disadvantages/ (9-sept-2016/1:30 PM)
8. Mohd. Ehmer Khan, "Different Approaches to White Box Testing Technique for Finding Errors" International Journal of Software Engineering and Its Applications,Vol. 5 No. 3, July, 2011
9. http://ezinearticles.com/?All-About-Code-Coverage---A-White-Box-Testing-Technique &id=1796983 (11-sept-2016/10:30)
10. http://www.testingbrain.com/tutorials/types-of-software-testing. html (12-sept-2016/3:00 PM)
11. Muhammad Shahid and Suhaimi Ibrahim, "A New Code Based Test Case Prioritization Technique" , International Journal of Software Engineering and Its Applications, Vol.8, No.6 (2014),pp.31-38. http://dx.doi.org/10.14257/ijseia.2014.8.6.03
12. Sudesh Kumar, Nupur Gupta," An Empirical study of Statement Coverage Criteria to reduce the test cases-A Review" International Journal of Application or Innovation in Engineering & Management, Volume 3, Issue 8, August 2014.
13. All about Code Coverage - A White Box Testing Technique by Yogindernath Gupta available at : http://ezinearticles.com/?All-About-Code-Coverage---A-White-Box-Testing-Technique&id=1796983 White-Box Testing by Laurie Williams published

AUTHORS PROFILE

Dr. Sasmita Padhy, belongs to Berhampur and borned on 10th June 1979. She received the B.E. degree in Computer Science from Utkal University, Odisha in 2001 and received M. Tech. in Computer Science from Biju Patnaik University of Technology (BPUT), Odisha in 2007. She also completed Ph.D. from Berhampur University, Berhampur, Odisha in 2012. Her area of interest is software testing, software engineer. She is currently working in Vignan Institute of Technology and Management, Berhampur as an Associate Professor in Computer Science and Engineering department. Before joining this college she had been worked in NIST(2010-2018), GIIT(2009-2010), JITM(2007-2009), GIET(2005-2007) and SMIT(2001-2005).

Dr. Satya Narayan Tripathy, received his M.C.A. and Ph.D. degrees in Computer Science from Berhampur University, Berhampur, Odisha, India, in the years 1998 and 2010 respectively. He is associated with Department of Computer Science, Berhampur University as Assistant Professor since 2011. He is member of several professional bodies such as, Life Member of CSI (Computer Society of India), Life Member of OITS (Orissa Information Technology Society). He has published numerous journal and conference papers in the area of data mining, malware analysis and computer security.

Er. Sisira Kumar Kapat, is Lecturer in Department of CSE, UCP Engineering School, Berhampur, Odisha, India. He has published several research papers in Journals of repute. He has also presented several research papers in several national and international conferences. He is a member of IEEE, member of IAENG, member of ICSES. The area of research is Malware analysis and detection, Computer Security, Data Mining.

Dr. Susanta Kumar Das, is Reader in Department of Computer Science, Berhampur University. He has published several publications in national and international Journals of repute over a span of 23 years in the area of software engineering, management information system and computer network security. He has supervised many doctoral and post-doctoral research works. He has felicitated award of honor by Department of Mathematics, Maharshi Dayanand University, Rohtak.