ISSN:0975-9646

N. R. Suganya et al, / (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 2 (5) , 2011, 2277 - 2279

A Survey on Detecting Domain Errors in Programs

N.R.Suganya[1], R.Venkateswaran[*2], K.Kathirvel[#3], M.Mohankumar[#4], G.Manivasagam[#5]

[1]*Karpagam University*
Eachanari, Coimbatore, Tamilnadu, India
[*2]*GR Govindarajulu School of Applied Computer Technology,*
Coimbatore, Tamilnadu, India
[#3,#4,#5]*Department of Computer Applications, Karpagam University*
Eachanari, Coimbatore, Tamilnadu, India

***Abstract*— There are many different techniques to overcome the problem of manual as well as automated test case generation. In this paper we try investigating how to overcome the problems and which way of proceeding will be a better one. For a program, there is large number of test cases generated. The first hurdle is the test case generation which plays a vital role in testing software. Why is it a hurdle? Generation of test case produce chaos for a tester as which part of a program is completed (either each and every line is completed or some part like loops or symbols or similar domains are completed). Hence a major challenge in this area is to generate a relatively small set of test cases by covering all the domains in a program. A test case should be generated right from the requirements gathering stage to wholly detect the domain errors. In this paper, we present an algorithm which will help in easy way of test case generation.**

***Keywords*— Test case generation, Domain errors, Test case description.**

I. INTRODUCTION

When a project is being started the first phase will be requirements gathering phase. If the project is done based on the V - model, then simultaneous process of testing can also be carried down. So it will be much advantageous to generate test cases at the starting stage of the project. Every domain can be isolated based on the number of variables, loops, decisions, links etc. When the isolation is completed test case would be generated without any chaos. We describe a strong strategy to detect domain errors for some other domains such as strings, function and more. This mainly detects all the errors in programs that guide testers to select a set of test points after test case generation. Since the program errors can be identified using white box testing, test cases generated will include the test points based on the binary pattern that is 1 or 0 (ON/OFF).

II. PROBLEM DESCRIPTION

Testing software should be preplanned and carried down in a strategic manner. Software engineers think they are angelic since they develop the code and the testers are to just point out errors. How much ever the cost is spent for developing, the same ratio is also spent for testing. Testing is not only to find faults but also to prevent software from bugs and make the end user happy. By means of testing, the software gets executed with the test data and the output is being examined. Main demerit is time consumption, chaos whether a line is completed or not, whether all domains are checked or not and more.

This paper tries to improve test performance as follows:

- Diagram to Implement Test Case Generation.
- Test Case Description.
- Test Case Generation Algorithm.

III. PROPOSED TECHNIQUE

A. Test Case Description

A format for test case generation is given below:

TC = {N_TC_ID, TCD, P_TC_ID, TI, TAO, TEO, FLAG}.

TC - Test Case
N_TC_ID - New Test Case ID
TCD - Test Case Description
P_TC_ID - Previous Test Case ID (if necessary)
TI - Test Input
TAO - Test Actual Output
TEO - Test Expected Output
FLAG - Red & Green

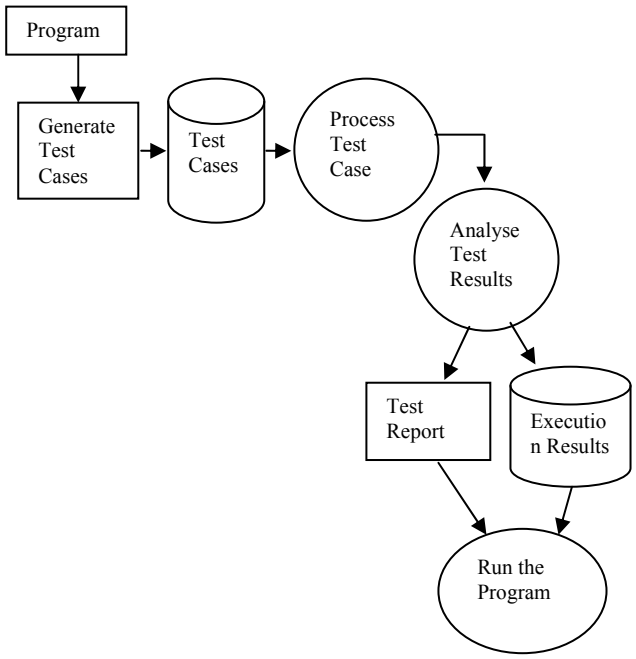
Based on this format given the test case can be generated. For a test case to be generated, first it should be given with an id. N_TC_ID can be used to check easily whether that particular test case has reached the expected output or not. P_TC_ID is applicable only if the test fails. TCD is the test case description of how that particular domain should produce the output. TI is the test input of how the tester should test inputting the values. TAO is the actual output got after the first level of testing is completed. TEO is the expected output of how the end user requires. FLAG acts as a Test Point.

Green => Actual Output = Expected Output => (OFF)

Red => Actual Output ≠ Expected Output => (ON)

B. Diagram to Implement Test Case Generations

This diagram given below gives an idea of how the test cases should be generated.



Consider a program. Once the program is given for testing first thing to do is generating test cases. After generation is completed, the test cases will be stored in a test case database. The test cases one by one will be processed by the tester. All the test cases will be completed and test results will be analysed. Based on the test results the test report will be produced of whether the expected output is achieved or not.

C. Test Case Generation Algorithm

An easy approach of generating test cases is by using BOUT algorithm. It is nothing but it fits any of the data structures for each domain in a program, which would be a stack or queue or linked list or anything else. Assume n parameters T_1 to T_n in which the target is set. It indicates different values in which particular test cases are selected to store in a different variable take $\{V_1$ to $V_n\}$. By using stack, the test case elements are bound together and when one parameter is checked it indicates stack empty and that variable gets released. So that part gets completed. It will be an iterative process.

Example:

Variable stack V1 to V5

D	V5
b	V4
a	V3
s	V2
S	V1

Flag

T5	Rule 1	
T4	Rule 2	
T3	Rule 3	
T2	Rule 4	
T1	Rule 5	

Test Case for Variable D

Rule 4	T2
Rule 5	T1

Target for Variable D not achieved

Hence we get V5 => D ≠ T1 & T2

After all the test case for Variable D is completed it gives stack indication (empty or full). Once the target T_n is achieved, the test cases for that particular target alone will be checked with the program and after modification the same process will be done. Hence it is an iterative process.

IV. Conclusions and Future Work

This technique is being used with stack. As a data structure is used it is advantageous to avoid mistakes in generating test cases. It can be implemented. This algorithm seems to be better when compared to other algorithms as testing can be completed successfully. But it will be a long process and so it is time consuming. Our future work is to implement the same algorithm after isolating the domains by means of CFG and implement with some other data structure which would be time consuming.

References

[1] Steven J.Zeil member IEEE, "Perturbation techniques for detecting domain errors" on IEEE transactions Vol 15, No.6, June 1989.

[2] P.Thevenod Fosse, H.Waeselynck and Y.Crouzet, "An experimental study on software structural testing : Deterministic versus random input generation" IEEE 1991.

[3] Faten H.Afifi, Lee J.White and Steven J.Zeil, "Testing for Linear errors in non linear computer programs" ACM transactions 1992.

[4] M.Roper, "Software testing – searching for missing link" Elsevier Science Information and software technology pg 991-994 1999.

[5] Masayuki Hirayava, "A Selective software testing method based on priorities assigned to functional modules" IEEE transactions 2001.

[6] D.Dvorak, R.Ragmussen, G.Reeves and A.Sacks, "Software Architecture Themes in JDL's Mission Data System" in proceeding of IEEE Aerospace Conference Mar 2000.

[7] Sudheendra Hongal and Monica S.Lam, "Tracking down software bugs using automatic anamoly detection" in ACM transactions ICSE may 2002.

[8] Nasha Miran, "Data Generation for Path Testing Software Quality" Journal Kluwer academic publisher's pg 121-136, 2004.

N.R.Suganya is pursuing her Ph.D at Karpagam University. She holds her MCA degree from Anna University, Chennai. Also completed her B.Sc Computer Science degree from Bharathiar University, Coimbatore. Being her research area is in software testing she also has interest on security oriented networks, Digital Processing and Software Engineering. She was working as a software engineer in UST Global for 2 years. And recently entered into academics. Organized workshops, FDPs, attended International conferences and published an International journal paper on Networks.

R.Venkateswaran is pursuing his Ph.D in Karpagam University. He Holds his M.Phil and M.C.A degree too. He is working as a assistant professor at GR Govindarajulu School of Applied Computer Technology, Coimbatore. He was working with Nehru College for past 5 years. His area of interest is Networking. He has published many international journals and has presented papers in Conferences. Also attended many FDP's.

K.Kathirvel, did his M.Phil from Karpagam University, Coimbatore. His area of research was Software Engineering. His professional qualifications include M.Sc from Bharathiar University. He was working as a software engineer in UST Global for 1 year. He has a working experience of 3 years in Karpagam University as Lecturer.

M.Mohankumar, did his M.Phil from Karpagam University, Coimbatore. His area of research was Software Engineering. His professional qualifications include MCA from Bharathidasan University. He has a working experience of $2^1/_2$ years in Palpap Software International Ltd as a test Engineer. He has a working experience of 3 years in Karpagam University as Lecture. Organized workshops, FDP,guest lecturing he was interested.

G.Manivasagam, did his M.Phil from Karpagam University, Coimbatore. His area of research was Software Engineering. His professional qualifications include MCA from Bharathiar University. He has a working experience of $2^1/_2$ years in Karpagam University as Lecturer.