International Journal of Emerging Technology in Computer Science & Electronics (IJETCSE)
ISSN: 0976-1353 Volume 20 Issue 2 – FEBRUARY 2016.

A DATA FLOW TEST SUITE MINIMIZATION AND PRIORITIZATION IN REGRESSION MODE

M.Vanathi[1]
Sona college of Technology
Sathya.vanathi11@gmail.com

J.Jayanthi[2]
Sona college of Technology
computer Science and Engineering

Abstract: **The objective of regression testing is to verify the new changes (addition /deletion) incorporated are implemented correctly. In next step, optimizing the number of test cases by making it effective and efficient. Here dataflow testing in regression mode is considered for verification and validation. Huge number of test cases must be minimized and prioritized based on the proposed algorithm. The program to be tested must be represented using control flow graph. The du and dc path must be identified. The recurrent definition and usage must be identified and test cases will be prioritized in proposed system using junit tool.**

Keywords

Test Minimization, Test Prioritization, Control Flow Graph,Data Flow Techniques.

I. INTRODUCTION

Software testing is essential phase in software engineering which is used to detect errors as early as possible to ensure that changes to existing software do not break the software and also used to determine the quality of software product. The main myth is good programmers write code without bugs.

Phases in a tester`s mental life can be categorised into 5 phases. They are

- Phase 0(Debugging Oriented)
- Phase1(Demonstration Oriented)
- Phase2(Destruction Oriented)
- Phase3(Evaluation Oriented)
- Phase4(Prevention Oriented)

Test Case is step by step description of the action that we do during the testing. Test Suite is the collection of test cases and it is way in which we are grouped the test case based on the module wise structure and rule module wise structure along with future.

Test prioritization is basic idea to group test cases based on some criteria and we can prioritize based on another set of criteria such as impact of failure ,cost to fix etc. We can prioritize the test cases by using methods like Cosine methodology, Greedy algorithm, prioritization metrics and measuring efficient etc. The Goals of prioritization are,

- To increase the rate of fault detection.
- To increase the coverage of code.
- To increase their confident in the reliability of the system.

TestCase minimization technique is used to find and remove the redundant test case from the test suite. Test cases became redundant because their input/output relation is no longer meaningful due to change in program and their structure is no longer in conformity with software coverage. It is yet another method for selecting tests for regression testing.

Regression testing is used to verify that changes work correctly and meet specified requirement. It is executed after defect fixes in software or its environment. Whenever the defects are done a set of test cases that are need to be rerun/ retesting to verify defect fixes are affected or not. Rerunning or retesting of all test cases in test suite may require an unacceptable amount of time. Minimizing the test case will overcome these difficult.

Data flow testing is based on selecting the path through the programs control flow in order to explore sequence of events related to status of data or variable or object. It flows on the points at which variable receives value and points at which values are used. It denotes each link with symbolslike d,k,u,c,p or sequence of symbols like dd,du,ddd,.etc that denotes sequence of operation. The data object state and usage are, Defined(d), Killed(k) and Usage(u).

The JUnit framework encourages developers to write test cases and hen to rerun all of these test cases whenever they modify their code. Whenever size of the code grows, rerun-all regression testing strategies can be excessively expensive.

II. RELATED WORK

This section consists of survey of test case prioritization and test case minimization techniques.

2.1 Minimize Test Case generation using Control Structure Method

Swathi.J.N and Sangeetha.S proposed an idea that software testing is the critical activity in any industrial strength software development process. As the software

grows in size, its complexity increases and testing becomes more difficult. Hence generating test cases manually makes more error so they propose an automatic generating of test cases using control structure method this tool aims to achieve coverage of a given structural code by including statement coverage, decision coverage , path coverage and branch coverage analysis and it also helps developer and tester to measure the effectiveness of test case generated using metric called "Test Effective Ratio".

To ensure that all statements have been executed at least once the Cyclomatic complexity is provided with number of tests. Complexity can be computed in ant one of the three ways,

- The number of regions of the flow graph(G) corresponds to the cyclomatic complexity V(G).
- V(G)=E-N+2,where E is the number of flow graph edges and N is the number of flow graph nodes.
- V(G)=P+1,Where P is the number of predicate nodes.

Test Effectiveness Ratio (TER) is classified by, dividing the Number of statements exercised by the test case by Total number of statements in the source code.

2.2 Generating Minimal Test Cases for Regression Testing

Sapna PG, and ArunkumarBalakrishnan proposed an idea that they generate the test cases from the specification of UML diagram and set of terminals are given as input to stenier tree algorithm and the minimal test case to check functionality. In stenier minimal tree problem, vertices are divided into two parts: Terminal and non-Terminal parts. The changed nodes are defined as terminal nodes to ensure inclusion in the test set. A lot of work is available for generating regression test cases both white box and black box strategies. A minimal set of test cases is generates as an indicator to the effectiveness of the change. Initial result shows that the method is applicable for quick testing to ensure that basic functionality works correctly.

UML activity diagram is developed using two types of nodes. They are action and control nodes. The action nodes consist of Activity, Call Behaviour Action, Send Signal and AcceptEvent. The control nodes consist of Initial, Final node, Flow-Final, Decision, Merge ,Fork and Join. This diagram is used to generate test cases.

The activity diagram is converted into Control flow graph. The weight for each edge is calculated by using measure where the calculation is measured based on the incoming and outgoing dependencies as given below,

$$\textbf{Weight(e)}=(n_i)_{in} \textbf{ X } (n_j)_{out}$$

Where$(n_i)_{in}$is the number of incoming dependency of node n_i and $(n_j)_{out}$ is the number of outgoing dependency of node n_j.

2.3 Minimizing Test Cases By Generating Requirement Using Mathematical Equation

Mamta Santosh and Rajvir sing propose idea of list of testing requirement for test suite and find the set of test case satisfy the testing requirement by requirement matrix, prioritization process of control flow graph, fault exposing potential value and test case requirement matrix formations. Test suite minimization techniques are used to remove redundant and obsolete test cases from test suite. Test case minimization approach can be considered as an optimization problem. Genetic algorithm can be used for minimization as it robust and provide optimized result. For applying genetic algorithm test case requirements relationships need to be transformed in mathematical model expressed in form of functions and parameters that optimize the model. At result the minimized test cases.

Here "Heuristic based approach selects test cases based up on the strategies of essential redundant and 1to 1 redundant strategies"

The fitness function is calculated by summing up the test case requirement matrices. Execution time has been applied for optimization.This approach reduced the test suite size and covered all requirements.

2.4 Test Suite Minimization by Greedy Algorithm

SriramanTallam and Neelam Gupta proposed an idea of test suit once developed is reused and update frequently as software will provide redundant is test case. Due to the resource and time constraints for re-executing large test suites, it is important to develop techniques to minimize available test suites by removing redundant test cases. Test suite minimization is NP complete. Here regularly measure the extent of test suite reduction obtained by algorithms and prior heuristics for test suite minimization. The same size or smaller size test suite selected than that selected by prior heuristics and had comparable time performance.

The concept analysis and test suite minimization consist of Object Implication, Attribute Implication and Owner Reduction. These all always preserve the optimality of solution for the context table. Then it uses greedy heuristic would be applied if the context table is not empty.

2.5 FIVE TECHNIQUES OF REGRESSION TESTING

GhinwaBaradhi and Nashat Mansour proposed an idea of comparing the 5 techniques of Regression testing – slicing., incremental, firewall, generic & simulated annealing algorithm. Comparison of these techniques is based on qualitative & quantitative- execution time, number of selected retests, type of testing, type of approach ,level of testing, precision, inclusiveness, user parameter. This comparison says that all algorithms are suitable for different requirements of regression techniques. The assessment is based on the following consideration, medium size modules

are important for assessment, since they are more realistic and execution time assessment is based on comparing the algorithms with each other.

This tends to indicate that the incremental algorithm has more favourable properties than the other four algorithms. The assessment is based on the following consideration,

- Medium size modules are more important for assessment,since they are more realistic.
- Since the test cases were manually developed, it was not possible to run experiments that were statistically highly-sound, especially for execution time.
- Execution time assessment is based on comparing the algorithms with each other.

To choose minimum number of test case and to perform fast regression testing these selections should be done genetically.

III. PROPOSED SYSTEM

Test suite minimization and prioritization is process of avoiding redundant and unwanted test suite. Till now minimization of test cases are done by applying several techniques like genetic algorithm, stenier tree algorithm, decision tree etc but in proposed system ,minimize test case is done by data flow testing techniques which is concern of regression mode.

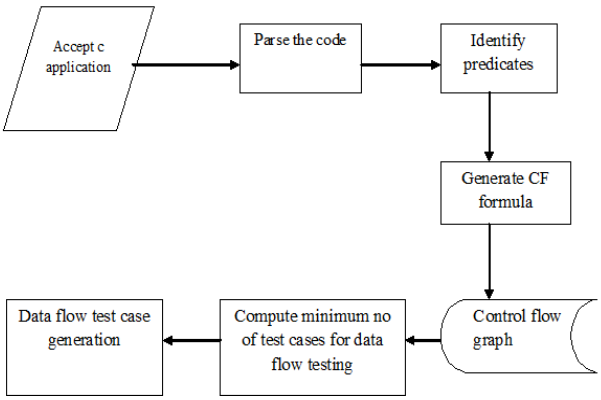

Fig: 3.1 System architecture of proposed system

Fig 3.1 describes the system architecture of proposed system. In this input is given as coding and output is in format of dataflow test case generation. Here also the processes of data flow testing techniques are used.

Here first java code get parsing by splitting the input into two things Split the input into tokens and then find the hierarchical structure of the input.Then they are converted into control flow graph(CFG). The test input data generation is done by two path Executable and Infeasible paths. Test case generation are generated with graph traversal. Test cases are specially written in JAVA and tested with JUnit test cases.

JUnit testing and prioritization

JUnit test cases are java classes that contain one or more test methods and are grouped into test suite.

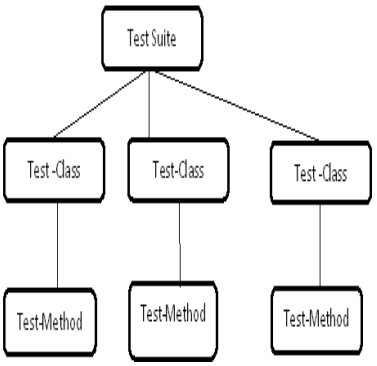

Fig 3.2 JUnit test suite structure

JUnit test classes that contain one or more test methods can be run individually or a collection of JUnit test cases can be run as unit. JUnit is designed tests to run as sequences of tests invoked through test suites that invoked test classes.

JUnit framework allowed us to achieve the following four objectives,

- Treat each Test Case Class as a single Test case
- Reorder the Test Case classes to produce a prioritized order
- Treat individual test method within test cases for prioritization
- Reorder the test methods to produce a prioritized order.

All test cases are run and executed individually with the JUnit test Runner in desired order. After the execution the result can be viewed using CLOVER views.Clover views consist of four views they are,

- Coverage explorer
- Test Run explorer
- Clover dashboard
- Test contribution

After all these reports of this also get by using clover views. By using this, data flow testing techniques are going to be tested.

IV. CONCLUSION

The proposed tool has been designed and developed with java code. The outcome of this tool is to

assist the tester to test the code in efficient manner. The test cases are tested and provide report about the test cases. Using this tool the test cases are generated and tested with its report. In this too all test are tested. JUnit tool help us to see the report in graphical way which is help full in easy understanding.

REFERENCE

[1] Adam Kurpisz,Samulileppanen, "On the sum of the squares hierarchy with knapsack covering inequality" ,arxiv: 1407. 1746v1 [cs.DS], 2014.

[2] Amrita jyoti,Yogesh Kumar and D.pandy "Recent priority algorithm in regression testing",International journal of information technology and knowledge management,volume-2,no.2,pp.391-394,2010.

[3] Alex kineer and Hyunsook Do "Empirical studies of test case prioritization in a JUnit testing environment"no.2,pp 1-12,2007.

[4] Bang ye wu: "A simple approximation algorithm for internal steiner minimum tree".CoRR, abs/1307,3822,2013.

[5] Baradhi.G and Mansour.N "A Comparative Study of Five Regression Testing Algorithm".Proceedings of IEEE international symposium on software testing and analysis,pp,143-152,1997.

[6] Jyoti and Kamna Solanki "A Comparative study of five regression testing techniques:ASurvey",IISN 2277-8616.IJSTR issues 8 aug 2014.

[7] R.Beena and S.Sarala, "Code Coverage based Test Caseselection and Prioritization",International Journal of software Enginneering& Application,vol.4,no.6,nov 2013.

[8] Sapna P.G and Hrushikes ha Mohanty "Prioritization of scenarios based on UML activity diagrams". IN 1st International Conference on computational intelligence, pages 271-276.IEEE computer society,2009.

[9] SwathiJ.N,Sangeetha.s "Minimal test case generation for effective program test using control structure method and test effective ratio"ijoca/0975-8887,48-53,2011.

[10] SriramanTallamand Neelam Gupta "A concept analysis inspired Greedy Algorithm for test suite minimization",ACM 1-59593-239-9/05/009.

[11] T.Rotho , "Directed stenier tree and the Lasserre hierarchy", CoRR, abs/1111.54-73,2011.

[12] T.Rembiszewski andBluemke "Dataflow testing of java programs with DFC",2000.