An Empirical Model of Regression Testing Using Hybrid Criteria

Posham Bhargava Reddy
JNTUA College of Engineering, Ananthapuramu, India

Abstract – Regression testing is an important of test case generation, coverage calculation, test case prioritization, test suite reduction (also called minimization), and test case selection are typically centered around a hybrid criteria that determines test case selection and test case executions. Regression test case selection describes three ways of combing hybrid criteria of integer programming i.e. max-max, max-min, min-min. Regression testing techniques are heuristics, however, and to properly evaluate their cost-effectiveness in practice, empirical studies are essential for retesting. Regression testing strategy usually refers to a rational way to define regression testing scope, coverage criteria, re-testing sequence and re-integration. Test case generation used an integer liner programming based on multi-criteria selection. Test case prioritization technique has used min greedy technique. The prioritization technique increase reliability of prioritization based on code coverage calculation. Test suite reduction applies an over writes of test cases. Test selection based on predetermined number of test cases in java file constrains of class, methods, lines, blocks, and instructions. Test cases has uniformly increases and decreases of code coverage percentile of test cases.

Index Terms – Regression Testing, generation, selection, prioritization, hybrid Test Case Selection.

1. INTRODUCTION

Testing activities occur after software changes or modifications on a program, Testing is mainly verifies or checking to errors, bugs and faults of programming languages or it is process of executing software programming languages to finding errors. Software Programming Languages (SPL) based on testing process determines or checks specification functionality, performance, verification, validation, quality and assurance. Effective testing delivers quality of products, i.e. user's needs, requirements, and expectations [10], [9], [8].

Today's world, many organizations will maintain the regression testing in the maintenance phase of errors and bug repositories to store the bunch of bug or error reports, which is useful for developers in the future of adding and increasing or decreasing the lines of code. A software developer working on a project they often visit the bug reports to understand the root causes of specific bugs and errors how previous actions has been taken to resolve the problem.

D. Cubranic et al. [12] proposed a search recommendation system which can help a developer to identify similar bugs and errors from bug repository. The developer needs to go through all recommended bug reports to identify the useful information relevant to what the developer wants modify the data. Trawling through a flood of data for all recommended bugs might consume more time. The developers still need to follow the monotonous process. G.C. Murphy. [12] Suggested that when bug report is resolved and closed, its respective author should write the summary of abstracts manually. This abstracted summary is helpful for developers who will visit the error report in the future and allow them to better understanding of the bugs and errors. However, in this method the developer who creates the abstracted summary wants to read the all conversations which are taking place between several stakeholders. So, it takes a lot of time to go through the flood of text and need human resources more. Because of the dynamic nature of bug report and requirement of human resources, it is not an optimal solution and not working in practice. Therefore, there is a need for automatic test case generation.

Ever line of software programming under code checks and verify-validate the test cases, whenever generates test cases there is no problem of the code. If may not possible to generate test cases there is problem on programming or code. The Code verifying line by line and checks the errors and fix that errors by automatically. Testing is reduces the risks of software because mainly risks comes under testing phase that why coding-testing phases are very difficult.

Software testing is important phases in Software Development Life Cycle (SDLC) and the software systems. It evaluates the capability or any attribute of the software program for achieving its desired results. Purpose of software testing can be quality assurance, verification and validation and reliability estimation. It can also be used as a generic metric.

The testing purpose is reducing the risks of the software. The unidentified factors are in the development, design and testing of new software can degrades the project quality of the software and delays it. By using a development cycle of testing and resolution you can recognize the level of risk, make well-versed decisions and ultimately reduce ambiguity and eliminate errors [10].

The automatic test cases generation is retests after fixes to the errors that ensure issues have been resolved before development of the progress. It has reduction of the test cases

in data, if the data has same the test cases are reduced in retesting of the automatic based on repeating tests in parallel areas and to fixed unexpected behavior of the data.

- Avoiding generation of the test cases that are repeated test cases in automatically using regression testing Based Approach.
- Generating the distinct test cases using prioritization based on different criteria.
- Improving code coverage based on hybrid criteria of test cases.
- Finally, an empirically model was performed to the test case generation and how it is prioritization based on section of the test cases of the regression testing understandable over the code coverage of the data repot.

2. RELATED RESEARCH WORK

G. Rothermel. [3] was different test case generation are defined and how to generate test cases of the data and prioritization of the formulas based on generation. The techniques are available to without modifications based on developed of test cases. The report of generating test cases of the model of the prioritization techniques are the test case generation models.

According to D. Cubranic [12] the summary of test cases produced for one by one test cases, hat contains performed of information of the test cases of the original data. Above definition states that a summary should not exceed 50% of the length of original texts. G.C. Murphy. [12] States that system generated summary must close to the human generated abstract summary. Automated text summarization aims to provide a condensed representation of the content according to the information that the user wants to get.

There are many techniques available for regression test case selection and prioritization. The empirical studies are reveal of different programs and techniques of relative performance of information; Regression testing is required consistent different programs and different criteria. Also many of these techniques use a single criterion and therefore the fault revealing phenomenon is probabilistic in nature. This limits their fault detection capability. The formation of regression testing selection, prioritization, and hybrid criterion approaches are very consistent to the previous techniques of the regression testing.

S.Sampath. [1] [2] has used Integer Linear Programming (ILP) with defined to developed error detection rates from previous runs for reduction of test suite and also combined different criteria in different ways such combinations were useful. The average percent of fault detection (APFD) was not increase number of test cases based on prioritization of test cases. Bryce. R has implemented the stand-alone criteria, second-order criteria ……and, n-order criteria were defined. The test cases has used in rank, merge, and choice based on implemented to the regression test cases, default the test cases generated with prioritization so code coverage has implemented of statement coverage, branch coverage and method coverage as usage based criteria. They evaluated to find APFD approach based on developed web applications so they didn't generate distinct test cases using different criteria and also not improving code coverage percentile of the methods. They gave differentiate the stand alone criteria versus hybrid criteria based on fault detections i.e. study of standard alone versus the hybrid criteria referees to advantage and disadvantages of the hybrid criteria, They had empirically measured, the methodology based on results are defined representation of multiple data distribution and cover maximum of the distribution.

Hyunsook Do.[5][9] defined test reduction using different criteria models based on implemented they proposed 3 policies that are :weighted, prioritized, and hybrid. The first policy is considered to weight of every object and considered all criteria with test suite reduction. Second policy is defined test case prioritization of the weighted and hybrid criteria. Third policy is hybrid, this divides single objectives …….and, n-order objectives with assign priorities. They methodology is seven programs from Siemens suite and additionally flex, logic blox, eclipse also from SIR [5]. They combined seeded and real faults in proposed system i.e. converged to solution quickly of most reduction problems and reduction with tie-breaking (RTB).

Elmar Juergens [6], [4], [3] methodologies were defined two that are: execution time and code coverage. First method is reduces execution time of test suite and second method is code coverage method of fault detections based on implemented with genetic algorithm approach. There experiments are defined two applications that are Gradebook and JDepend .so proposed with their time –aware prioritizations outperform techniques.

3. REGRESSION TESTING USING HYBRID CRITERIA

3.1. Test case generation

In Generation, the test cases for the program to be tested are generated using the general random test case generation method. The generated test cases are then executed against the program and their execution details are captured. The execution details include execution time of the test case, code coverage data of the test case, and the detected faults.

Input: Software program

Output: Test suite and Execution details of test cases

Methodology: Test case generation technique

Steps:

1. Select the program for test case generation.
2. Consider the program line by line
   2.a If a software element
   Then generate and save the value
   Else go to next line
3. Execute the test cases against the program.
4. Record execution details

3.2. Coverage calculation

Coverage module uses the code change data and change impacted code data to choose all the test cases that come in the limit of the changed code area. Then it determines the sum of coverage, max-min criterion and the coverage size of each test case.

Input: Code change data and test suite

Output: Possible set of test cases and its Coverage data Methodology: EMMA code coverage tool

Steps:

1. Fetch the code change data.
   1.a Compare P and P' line by line
   1.b If changed
   Then add to code change data
   Else Go to next line
2. Select the possible test cases with respect to code change data.
   2.a Compare test case with code change data
   2.b If test case covers
   Then select the test case
   Else Go to 2.a
3. Determine coverage of test cases with EMMA.

3.3. Integer programming

This module formulates an Integer programming problem using the coverage criteria from the coverage module. An optimal solution is formed with Linear programming technique and a solution point is obtained.

Input: Coverage data of set of test cases

Output: A sub-optimal solution set of test cases

Methodology: Simplex method for linear programming

Steps:

1. Formulate Integer linear programming with coverage data.
   1.a Calculate fault detection capability

$$d_m(S) = \sum_{sn \in S} \delta m(sn)$$

   1.b Formulate D-functions for coverage criteria

$$D_{sum}(S) = \sum_{am \in A} wm\ dm(S)$$

$$D_{min}(S) = \min w_m d_m(S)$$

2. Solve IP problem using linear programming.

3.4. Test Case Selection

In this module a voting scheme for selected the final set of test cases from the solution set obtained from the integer programming module. For this, some elite subset of the solution points are decided. These points vote for the test cases to be included in the final solution, which are the selected test cases.

Input: The sub-optimal solution set of test cases

Output: Final solution set of test cases

Methodology: Voting scheme

Steps:

1. Select elite points from the sub-optimal solution set.
   1.a Get upper bound value of D_{min}
2. Apply voting scheme.
   2.a Elite points vote for favored test cases
   2.b For each test case
   If (vote >2)
   Then add to final solution
   Else
   Ignore the test case
3. Get the final solution set

3.5. Test Case Prioritization

The selected test cases from the selection module are prioritized here. The technique used is an algorithm called MIN-greedy.

Input: Final solution set of test cases

Output: Prioritized set of test cases

Methodology: MIN-greedy prioritization technique

Steps:

1. Fetch the coverage data and change degrees of software elements A.
2. Get the set of bottleneck elements
   2.a If current coverage of A = minimum
   Add the elements to the set
   Else go to next element
3. For each test case
   3.a Find the number of bottle neck elements covered.
4. Sort the software elements
   4.a If current coverage(Ai) < Current coverage (A_{i+1})

Then add Ai to first position
Else add Aj to first position
Increment I;
Go to 4.a

5. Get the test case that covers most number of test cases
5.a Add the test case to final list
5.b If coverage (Ti) = coverage (Ti+1)
Then test case with largest coverage for first element added to list
Else
Take the second element in the ordering
Else
Test case with largest index added to list
6. Get the final solution set.

4. METHODOLOGY AND DEMONSTRATION

Regression testing using hybrid criteria has used in algorithms of test case generation, coverage calculation, integer programming, and test case selection

- case generation technique

A general test case generation technique is applied to the input program. The test cases are generated and are stored for further selection and prioritization.

- Integer programming based multi-criteria selection technique.

The multiple criteria as input and using it an integer linear programming problem is formulated. The multiple criteria include the coverage data of the test cases which covers the changed area of code. The coverage data id used to formulate the integer programming problem. The IP problem is solved using linear programming technique since IP is a NP hard problem and a sub-optimal solution set is obtained [3]. To select the optimal sets of test cases, a voting scheme is used. First some elite points are selected from the sub-optimal points. Each facade of a test case is one of the "elite" result points is a vote for that test suite of test cases. After the voting is achieves, the L test cases that have the upper most votes are selected, as long as they have at least two votes.

- Min-greedy (GMIN) algorithm for prioritization.

This technique prioritizes the selected test cases on the basis of coverage data and change degree of the code. The MIN-Greedy algorithm takes two sets of inputs: coverage data, and the priority measure for software elements. The software elements with minimum current coverage are found out and the test cases that cover these elements are identified. Then the software elements are sorted based on ascending current coverage. The algorithm adds test cases to the result set one by one in iterations. Every iteration has maintains the result set based algorithm takes the adding the data of the every iteration. It keeps of the code coverage percentile of the every program software programming of the code after adding test cases so far obtained the result set.

4.1. DEMONSTRATION

The regression testing using hybrid criteria firstly generated test cases with code coverage approach .the generation technique to take input has a programming and gives the test suite with code coverage percentile .it captured the execution of program profile, if the data has changed again retesting with changed data and previous data based on code coverage technique

Consider any java program P and programming test cases T={t1,t2,t2,t4……tn}.programming contains some parameters that are classes, methods, packages and lines this are also called elements A={a1,a2,a3…an}. The test suite has n number of test cases has generated with code coverage percentile in IP (integer programming). Function F contains some faults that detects D-function after achieves a efficient results. D-function defined capability of detecting faults and code coverage of elements is S

Assume that let $d_m(S)$ capture the capability of detecting faults and code coverage of elements is S as applied to $a_m \epsilon A$.

$$d_m(S) = \sum_{sn \in S} \delta m(sn) \quad (1)$$

Where δ_m (s_n) captures the test case of effectiveness $sn \epsilon S$ as functional to element $a_m \epsilon A$.

D-functions has detects sum of coverage and max-min and max-max criteria are as follows.

Files in java	Class	Methods	lines	blocks	instructions
Student.java	100	57.1	88.5	76.6	92.8
Sort file data	100	75	87.1	83.9	92.5
Jaccard distance	100	75	79.4	80	86.2
Hybrid criteria	100	69.6	87.3	78.3	92.2

Table1: percentile of code coverage in hybrid criteria

NOTE: All values are percentile

$$D_{sum}(S) = \sum_{am \in A} wm\ dm(S) \quad (2)$$

and

$$D_{min}(S) = \min w_m d_m(S) \quad (3)$$

Where w_m ,d_m s are a set of weights emphasize elements of software that is more important to the detection faults process. W_m s account for the changes of since last release.

Hybrid criterion of regression testing using code coverage checks blocks, classes, lines, methods and instructions. The above equations 1 is captures the test case of effectiveness ,equation 2 is detects sum of coverage and max-min and max-max criteria and equation 3 is refers to emphasize elements of software. The above equations based on developed integer linear programming technique has defined .IP problems are formulated to results of solution set obtained. The final test suite is obtained solution set.

The MIN-greedy algorithm technique used for prioritization of test cases and code coverage percentile of the test cases [4]. The below fig1defind phase I and phase II, The phase I describes generating test case with the adding data or file to test based on previous data. The phase II describes selection test case and combinations of hybrid criterion based on generate test case with efficient test cases. This test cases has efficient because the code coverage technique used. Phase I is generating test case given step by step procedure to adding data or files of max-min criterion of the IP and comes the results adding the solution set of test suite. Phase II selects the combinations of the test cases with code coverage technique of MIN-greedy algorithm. That based generating test cases of coverage percentile comes in the output of the test cases.

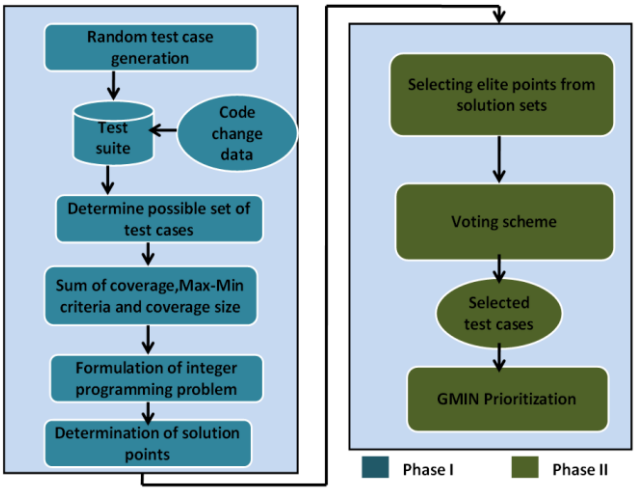

Fig 1: Test cases generation based on code coverage criteria

5. EXPERIMENTAL RESULTS

Regression testing using generate test cases, test case selection, prioritization with code coverage and integer linear programming .Test case generates automatically in adding data of the previous data. Test cases selection takes combinations of elements i.e. classes, methods, blocks, lines or statements branch based on code coverage percentile of the user programmers that is java programming .if adding data in previous data again generating test cases with code coverage percentile .Table 1 shows the percentile of the code coverage and generated test cases.

The above table and below fig 2 graph has represented in code coverage percentiles of the test case generated with criteria.

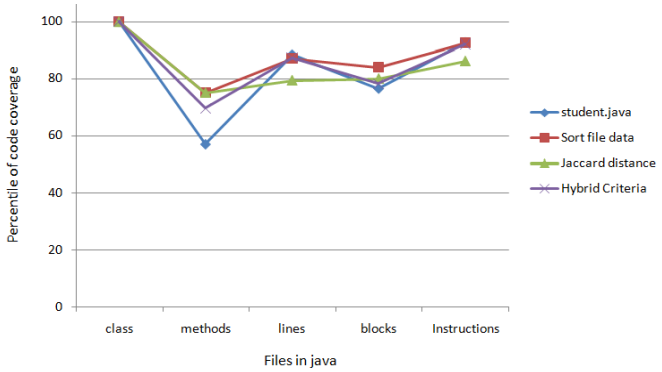

Fig 2: code coverage percentile of test cases

6. CONCLUSION AND FUTURE WORK

Regression testing using hybrid criteria describes code coverage percentile with generating test cases. The prioritization of test case and test case selection with hybrid criteria. The hybrid criteria means combinations of classes, methods, blocks, and lines. This combinations are gives the more percentile then previous methods. Methodology is determine test cases with EMMA.

Regression testing combinations are through uniform based on empirical studies of generating test cases generate. Generating test cases typically there policies, that are coverage calculation, test case selection, and test case prioritization. Test case prioritization results are prioritize test cases, selection is selects combinations that are classes, methods, blocks, lines…etc. the coverage calculation depends on classes, methods, statements, blocks. This combinations based on generates code coverage percentile.

Regression testing future work anticipate many techniques and many technology based develop the increases percentile of the code coverage. Different ways to take different combination based on generate test cases with coverage percentile of the java programming of the regression testing. If any language to generate test cases in different combinations of the code coverage techniques. it increases more percentile of the code coverage and better than previous work.

REFERENCES

[1] Sredevi Sampath, Renee Bryce, Atif M. Memon "A Uniform representation of hybrid criteria for regression testing" IEEE Trans. Software Eng., vol. 39, no. 10, pp. 1326-1344,Oct.2013.

[2] S. Mirarab, S. Akhlaghi, and L. Tahvildari, "Size-Constrained Regression Test Case Selection Using Multi-Criteria Optimization," IEEE Trans. Software Eng., vol. 38, no. 4, pp. 936-956, July/Aug. 2012.

[3] G. Rothermel, R.J. Untch, and C. Chu, "Prioritizing Test Cases for Regression Testing," IEEE Trans. Software Eng., vol. 27, no. 10, pp. 929-948, Oct. 2001.

[4] Siavash Mirarab, Soroush Akhlaghi, Ladan Tahvildari, "Size-constrained Regression Test Case Selection Using Multi-Criteria Optimization". IEEE Transactions on Software Engineering, June 2011.

[5] Hyunsook Do, Siavash Mirarab, Ladan Tahvildari, Gregg Rothermel, "The Effects of Time Constraints on Test Case Prioritization: A Series of Controlled Experiments". IEEE Transactions on Software Engineering, volume 36, Sept.-Oct. 2010 **Page(s):** 593 – 617

[6] Elmar Juergens, Benjamin Hummel, Florian Deissenboeck, Martin Feilkas, Christian Schlögel, Andreas Wübbeke, "Regression Test Selection of Manual System Tests in Practice", 15th European Conference on Software Maintenance and Reengineering, **Page(s):** 309 – 312, March 2011

[7] Songyu Chen, Zhenyu Chen, Zhihong Zhao, Baowen Xu, Yang Feng, "Using Semi-Supervised Clustering to Improve Regression Test Selection Techniques", Fourth IEEE International Conference on Software Testing, Verification and Validation, **Page(s):** 1 – 10, March 2011.

[8] Alok Ranjan Pal, Projjwal Kumar Maiti and Diganta saha, "An Approach To Automatic Text Summarization Using Simplified Lesk Algorithm And Wordnet," International Journal of Control Theory and Computer Modeling(IJCTCM), Vol.3, no.4/5, 2013.

[9] X.Wang, L.Zhang, T.Xie, J.Anvik and J.Sun, "An Approach to Detecting Duplicate Bug Reports Using Natural Language and Execution Information," Proc. 30th Int'l Conf. Software Eng. (ICSE '08), pp. 461-470, 2008.

[10] S. Breu, R. Premraj, J. Sillito, and T. Zimmerman, "Information needs in Bug reports: Improving Cooperation between Developers and users," Proc. ACM Conf. Computer Supported Cooperative Work (CSCW '10), pp. 301-310, 2010.

[11] R. Lotufo, Z.Malik and K.Czarnecki, "Modelling the 'Hurried' Bug Report Reading Process to Summarize Bug Reports," Proc. IEEE 28th Int'l Conf. Software maintenance (ICSM '12), pp. 430-439, 2012.

[12] D. Cubranic and G.C. Murphy, "Hipikat: Recommending Pertinent Software Development Artifacts," Proc. 25th Int'l Conf. Software Eng. (ICSE'03), pp. 408-418, 2003.