A Survey on Techniques, Frameworks and Metrics in Regression Testing

T.R. Anand, Ph.D. Research Scholar,
PG and Research Department of Computer Science,
Dr. N.G.P. Arts and Science College,
Coimbatore, India
tranand1983@gmail.com

Dr. B. Rosiline Jeetha, Head,
PG and Research Department of Computer Science,
Dr. N.G.P. Arts and Science College,
Coimbatore, India
jeethasekar@gmail.com

Abstract—**Regression testing aims to check program correctness even after it was modified. Whenever new features are added to an existing software system, the new features along with the existing features should be tested to ensure that their behaviors are not affected by the modifications. Regression test selection (RTS) techniques reduce the number of regression tests to execute using some selection criteria. Test Case Prioritization focuses on increasing the fault detection rate. A detailed survey on the various techniques, frameworks and metrics for efficient regression testing is presented.**

Keywords-software testing, regression testing, regression test case prioritization

I.INTRODUCTION

Software testing is a process, in which the software system is executed to locate all the faults which cause failures, and also to fix the faults detected, so that the quality of the software is improved. [1]. Regression Testing, a type of software testing, verifies the previously developed and tested software performs correctly even after it has been enhanced. The regression testing uncovers software bugs or regressions. A software regression is a bug which makes one or more features stop functioning or malfunctioning as anticipated after a system upgrade, system patching etc. Regression Testing should be done on scenarios such as: when requirements are changed and the code is customized according to the new requirement, when new feature is added to the software, and when performance issues to be fixed. The purpose of regression testing is to make sure that the new changes have not introduced new faults, and also to determine whether a change in one part of the software affects other parts of the software. Defects found during the regression testing are costly to fix. Regression testing includes rerunning previously completed tests to check whether the behavior of the program has changed and the faults previously fixed have re-emerged. Regression testing is done with full or partial selection of previously executed test cases to make sure the existing functionalities work fine and faults previously fixed have not re-emerged. Regression testing can be efficiently done systematically by selecting the minimum set of suitable tests needed to sufficiently cover a particular change.

II.MOTIVATION AND PROBLEM STATEMENT

The major testing challenges in doing regression testing:

- The entire regression test suite cannot be executed because of the increased size of the test suites on successive regression runs. The large test suites make great impact on the time and budget constraints of the regression test execution.
- One of the great challenges in regression testing is to achieve maximum test coverage through minimized test suite.
- Determining the frequency of Regression tests, i.e., after every change or every build update or after a bunch of bug fixes, is also a challenge.

Regression testing, a cost and time consuming testing process, needs effective test case selection, minimization and prioritization strategies.

III. OBJECTIVE

The objective of this work is to make a survey on various strategies and metrics used in the process of Regression Test case Optimization in terms of test case selection, minimization, and prioritization.

IV. RELATED WORKS

Schwartz and Do, 2016 proposed a new fuzzy AHP (Analytical Hierarchy Process) strategy[2] to deal with the imprecision in decision makers' judgments in the AHP method. The authors also proposed a fuzzy expert system (FESART) to minimize the time needed by the decision makers does not require pair wise comparison. A new strategy WSM (Weighted Sum Model) to investigate the effectiveness of a simple, low-cost strategy for ATP has also been proposed in the work. The subject programs ant, xml-security, jmeter, nanoxml, and galileo are taken from SIR infrastructure. The cost-benefit has been evaluated in the strategies proposed. The strategies proposed in the work provide cost benefits than other strategies. Overall, the proposed FESART strategy provides most cost benefit among all the proposed strategies.

Risk-based test case prioritization using a fuzzy expert system[3] have been proposed by Hettiarachchi et al., 2016. In the proposed approach, the risks are estimated by associating with the requirements, the risk exposure for requirements are calculated, the risk exposure for risk items are calculated, and the requirements and test cases are prioritized. Requirements complexity (RC), requirements size (RS), requirements modification status (RMS), and potential security threats (PST) are used as risk indicators in estimating the risk. The proposed approach has been evaluated with in terms of Average Percentage of Fault Detection (APFD) and Percentage of Total Risk Severity Weight (PTRSW). Two subject programs; iTrust (version 0, 1, 2, and 3) an open source application by Realsearch Research Group , North Carolina State University, and Capstone, an industrial application developed collaborated with North Dokata State University has been considered to examine the effectiveness of the proposed approach. The proposed approach detects faults earlier and was effective in earlier fault detection in the high-risk system components.

Lin et al., 2014 evaluated the chance of each test case being replaced by others during test suite reduction using a cost-aware test case metric, called Irreplaceability and its extended version EIrreplaceability [1]. A cost-aware framework has been proposed integrating the concept of test irreplaceability into various well-known test suite reduction algorithms such as Greedy, GRE, and HBS. The authors reviewed existing Coverage and Ratio metrics implemented in Greedy and GreedyRatio algorithms respectively. GreedyRatio algorithm outperformed Greedy algorithm in terms of execution cost. The authors found in some circumstances that the metric Ratio does not lead to the test suite with the lower execution time. The authors propose GreedyIrreplaceability algorithm with Irreplaceability as the cost-aware metric and the time complexity is evaluated as O(m.n.min(m,n).k) and is better than the worse-case time complexities of Greedy and GreedyRatio. A new GreedyEIrreplaceability algorithm which incorporates the cost-ware metric EIrreplaceabity has been proposed by the authors. The time complexity of GreedyEIrreplaceability algorithm is O(m.n.min(m,n).k) and is been found that it is same as GreedyIrreplaceability algorithm. In worse case, it has been found that the GreedyEIrreplaceability algorithm is not much more costly than Greedy and GreedyRatio. The cost aware metrics Ratio, Irreplaceabilty, and EIrreplaceabilty has been incorporated into GRE and HGS algorithms. The subject programs from Software-artifact Infrastructure Repository have been used in the experimental evaluation.. From the experimental results, it has been found that among the Greedy algorithms considered for the evaluation, the GreedyIrreplaceability algorithm provide best cost reduction capability in which the SCR (Suite Cost Reduction) ranges from 90.04% to 97.64% for the subject programs. Among the GRE algorithms considered for the evaluation, the GREEIrreplaceability algorithm has been found providing the best reduction capabilities in which the SCR ranges from 88.13% to 96.75%. Similarly, among the HGS algorithms considered for the evaluation, the HGSEIrreplaceability algorithm has been found providing the best reduction capabilities in which the SCR ranges from 88.71% to 97.13%. Overall, it has been found that the metric EIrreplaceabilty was efficient than the Ratio metric for the considered traditional algorithms.

In the research work by authors Panichella et al., 2013, DIversity based Genetic Algorithm (DIV-GA) [4] based on the mechanisms of orthogonal design and orthogonal evolution that increases diversity by injecting new orthogonal individuals during the search process has been introduced. A set of 11 open-source and industrial programs from the Software-artifact Infrastructure Repository: space, an interpreter for Array Description Language, by European Space Agency; six GNU open-source programs bash, flex, grep, gzip, sed and vim; sprinttokens, printtokens2, schedule, and schedule2 from Siemens suite have been have been used to evaluate the proposed algorithm. The metrics such as increased code coverage capability, decreased execution cost, and increased past fault coverage were considered in the evaluation. The LOC of the subject programs ranges from 374 to 1,22,169, and the test case count ranges from 215 to 13583. The proposed DIV-GA has been compared with other algorithms such as greedy algorithms and the island version of NSGA-II (named vNSGA-II). The solutions provided by DIV-GA detects more faults than the other algorithms, while keeping the same test execution cost. The sub-test suites generated by DIV-GA were able to expose more faults at same level of execution cost than the sub-test suites obtained by both the additional greedy algorithm and vNSGA-II.

Heuristic based-Regression Test Prioritization [5], a regression TCP technique have been proposed by Panigrahi and Mall, 2014. The proposed technique was based on the analysis of a dependence model for object-oriented programs. An intermediate dependence model of a program was constructed, and is updated with changes when the program is modified. The affected node in the model has been determined by constructing the union of the forward slices which corresponds to each unchanged model element. Control and data dependencies are represented in the dependence model. The test cases were selected based on a study of control and data dependencies as well as dependencies arising from object relations. The test cases were then prioritized by H-RTP by assigning decreasing weights to affected nodes in the ESDG model. The authors also developed Heuristic-based TEST Prioritization (H-TESTPrio) tool which implements H-RTP. The subject programs considered in the experimental evaluation are Elevator Control ,Cruise Control, Binary SearchTree, AutomatedTellerMachine, Power Window Controller, Ordered Set, and Vending Machine. The LOC and test suite size of the subject programs ranges with 229 to 943 and 17 to 42 test cases respectively. The APFD for the proposed approach for the subject programs ranges from 88.5 to 92.3, where as the APFD for Smith et al.'s approach and for S-RTP for the subject ranges from 60.4 to 68.7 and 79.6 to 86.2 respectively. The experimental result of the work reveals that the APFD metric has been increased by 9.01% for H-RTP than the related approach.

For test case prioritization, Malhotra and Tiwari, 2013, proposed a framework[6] with a GA based test case prioritizing tool, the Modified APBC Metric (APBCm) and the Additional Modified Lines Of Code Coverage (AMLOC) graph as major components. APBCm have been used as fitness evaluation function in the Genetic Algorithm to evaluate the effectiveness of a test case sequence. The APBCm metric have been used by the prioritization tool to compare between two permutations and to decide the better permutation candidate and carried to next generation of the population. The simplified version of the original Triangle program with 10 test cases has been used to evaluate the proposed framework. The original subject program has been modified with few features. The prioritized test case sequence has been generated using the proposed tool with APBC and APBCm as fitness function in the Genetic algorithm. From the experimental results, the coverage rate using APBCm has been observed as 0.80285, whereas 0.79285 using APBC, which reveals that the APBCm metric is efficient than APBC.

A multi-objective test case prioritization strategy[7] have been formulated by Sun et al., 2013, to combine event coverage and statement coverage for GUI applications. The proposed strategy takes cost into consideration. In their work, fixed time interval between the events has been set when test cases were created, and the number of events has been used to estimate test case cost. As a result, a new fitness function for multi-objective test case prioritization has been proposed. To determine the high prioritized test case to be selected, the fitness function was defined to evaluate test case fitness with respect to the intention of different strategies. Two popular open source GUI applications, Crossword Sage and OmegaT obtained from SourceForge have been tested. The applications were tested to evaluate the Average Percentage of Faults Detected. The experimental evaluation reveals that the proposed multi-objective strategy performed better when compared with single objective strategies (statement coverage and event coverage).

Kim and Baik, 2010 proposed Fault Aware Test Case Prioritization Technique[8], FATCP, considering coverage and historical fault information by integrating with fault localization technique. The proposed approach uses the historical fault detection details of test cases, to adjust the priorities of fault-found test cases while preserving test cases with high coverage in high priority. Sample programs from SIR: tcas, totinfo, schedule, schedule2, printtokens, printtokens2, replace and space, with LOC varying from 138 to 6218 and test case pool size varying from 1052 to 13585 have been experimentally evaluated in terms of Average Percentage of Fault Detection with respect to the faulty versions. From the experimental evaluation, it has been found that the proposed FATCP technique reduces the total cost of executing entire test suite and earlier faults detection compared to the prior coverage-based techniques.

Bharti Suri and Shweta Singhal, 2011,[9] validated the test case prioritization using Ant Colony Optimization technique proposed by Singh et al, 2010 and implemented in the work done by Suri and Singhal. Seven C++ Programs; CollAdmission, HotelMgmnt, triangle, quadratic, cost_of_pub, calculator and prev_day , and one java program railway_book have been considered for experimental analysis. Fault seeding technique has been used to generate 5 to 10 modified versions and black box test cases for the programs. The LOC, number of versions, and number of test cases of the subject programs varies in a range of 31 to 666, 5 to 10, and 5 to 26 respectively. The total test suite execution time of the subject programs ranges with 49.84 to 468 seconds. Their results show that the proposed test suite selection and prioritization approach reduced the size of test suite, considerably reduced the execution time, the correctness gained has been very high for most of the test programs, and the faults were discovered earlier by the ordered test suite. All their observations implied that the ACO technique in prioritization and selection gave better results at higher TC values with minimal effect of extra time taken by the algorithm.

Indumathi and Selvamani, 2015 proposed 4 algorithms [10], one to derive dependency structures among test cases, another for level ordering arrangement, and two more algorithms Weighted_DFS(root level) and WDFS_visit(root to leaf level) to prioritize the test cases automatically. The proposed approaches have been evaluated with seven sample programs: print_tokens, print_tokens2, replace, schedule, schedule2, tcas and

tot_info, from Siemens test suite (SIR). The LOC, test size pool, number of versions, and number of functions of the sample program varies in a range of 173 to 726, 1052 to 5542, 7 to 4, and 7 to 21 respectively. The APFD of the sample programs have been experimentally evaluated. Their results revealed that the proposed techniques provided a better solution to the test case prioritization problem than the existing algorithms.

An approach for selecting regression test cases in the context of large-scale database applications is presented by Rogstad et al., 2013. Specification-based[11], a black box approach which relies on classification tree models to model the input domain of the System Under Test has been focused to obtain a more realistic and scalable solution. Similarity based test case selection which is cost effective has been incorporated with partition based approach to refine regression test selection. An experimental study has been made to evaluate the best fault detection rate and selection execution time for the evolutionary and greedy selection algorithm with the best suited similarity function among Euclidian, Manhattan, Mahalanobis and NCD. The selection strategies; random partition based, similarity partition-based, and pure similarity based, has also been evaluated in terms of best fault detection rate and selection execution time. The experiment has been conducted on a large and critical database application of the Norwegian tax department. The subject test suite for the experiment contains 5,670 test cases, splited across 130 partitions, and is based on actual data from the production environment in the SOFIE project. Due to the practical consequences of the variation in selection execution time, the fault detection rate has been used as the main criterion for comparison. Combined Mahalanobis similarity function and the 1 + 1 EA algorithm have been proved to be the efficient in terms of fault detection rate. And the Similarity partition based test case selection has been found far better fault detection rates compared to a random selection of test cases.

Modified Cost-Cognizant Test Case Prioritization (MCCTCP) [12] was proposed by Huang et al., 2012. From the historical information repository, MCCTCP acquires the test costs, the fault severities, and the detected faults of each test case from the latest regression testing. The test cases are prioritized by providing the data acquired from the historical information repository as input data to Genetic Algorithm to find out an order with the greatest rate of "units of fault severity detected per unit test cost". The historical information has been used in calculating the fitness value according to the cost cognizant metric, APFDc. MCCTCP utilizes the historical information from historical information repository, which is stored as previous execution result of each test case, to schedule the test cases. Two open-source tested programs of 5 versions; flex and sed, obtained from the Software-artifact Infrastructure Repository (SIR) were used to evaluate APFD. The faults are seeded by hand in each version. The LOC and number of functions, faults and test cases of the versions of the subject program sed are in the range 6671 to 11990, 120 to 285, 4 to 6, and 360 respectively. The LOC and number of functions, faults and test cases of the versions of the subject program flex are in the range 12424 to 14240, 149 to 167, 5 to 13, and 525 respectively. The proposed technique has been comparatively evaluated with other techniques such as random, optimal, hist-fault-detected, GA-hist, Hist-value, GA-fun, Total-fun, and cc-total-fun. The evaluation results of the work states that the proposed method has effective the fault detection than the three coverage-based and two history-based techniques. The proposed technique performed well when test case costs and fault severities were equal. The proposed technique provided higher efficiency than other GA based prioritization technique, because the number of faults was far fewer than the number of functions.

A fuzzy expert system [13] has been proposed by Xu et al., 2014, for test case selection with the constraint that source code analysis is not available. The potentially critical test cases for system test is identified by the proposed fuzzy expert system by associating knowledge represented by customer profile, analysis of the results of the past test case, rate of system failure and change in system architecture. Case studies have been made with data collected from two very large systems, a mobile communications infrastructure system and a mobile communications device, from the telecommunications industry. The proposed fuzzy expert system developed test plans for the two large real-world software systems. The empirical results revealed a significant improvement on the defect detection rate by the proposed fuzzy expert system technology. And also earlier defect detection has been additionally observed in the testing phase with a fewer number of test cases. The experimental results of the proposed work states that it is appropriate for solving inaccurate and subjective problems encountered during the system-level test case selection process.

Haidry and Miller, 2013, presents test suite prioritization techniques [14] DSP volume and DSP height for open dependency structures, and DSP sum, DSP ratio, and DSP sum/ratio for closed dependency structures which provides weights to paths in the dependency structure, rather than individual test cases. The above techniques prioritize the test cases based on the dependency structure of the test suite. Calculating Test case ordering was done through weighted depth-first search algorithm; where the weights are defined by the graph coverage values. Entire test suite prioritization was done using a greedy algorithm which selects the next path as the path with the highest priority. Six systems: Elite, GSM, CRM, MET, CZT, and Bash were used to empirically evaluate the strength of the proposed techniques, measured by the average fault detection rate, comparatively with randomly generated, greedily generated, and the untreated test suites used by test engineers. The result shows that the proposed techniques outperformed the random and untreated test suites, but not as efficient as the greedy test suites. For open dependency graphs, the proposed techniques achieved better APFDs for most experiments. For open dependency structures, the improvement was stated as lower execution cost. For closed dependency graphs, the proposed techniques achieved improved APFDs than total function coverage.

A multicriteria nongreedy optimization approach [15] is proposed by Mirarab et al., 2012, to solve size-constrained regression test case selection. A Size-constrained Regression Test Selection (SRTS) problem has been introduced to select the subset of the test suite with a known number of test cases. A new criterion maximizing the minimum sum of coverage (max-min criterion) across all software elements has been introduced for regression test suite selection and prioritization. An Integer Linear Programming (IP) problem has been formulated combining the max-min criterion and a second criterion of sum of coverage as two distinct optimization criteria. An approximate and suboptimal mathematical solution has been proposed to the IP problem that results in a set of solution points based on different weightings of the two objectives. A voting mechanism combined the set of solution points into one final solution. A greedy algorithm prioritized the selected subset of test cases. The IP- based multi-criteria non-greedy selection (IP) techniques proposed has been compared with other techniques such as Greedily optimizing minimum coverage(GMIN), IP+GMIN, Method-level total coverage(MTC), Method-level additional coverage (MAC), BN-based approach (BN), BN-based approach with feedback (BNA), Walcott et al.'s Time Aware Prioritization approach (WTA). Five open source Java programs: Ant, Nanoxml (Nano), Galileo, Xml-security (Xml), and Jmeter, obtained from SIR have been used for evaluating the proposed approach. The number of versions, classes, test cases, mutants, and KLOC of the subject programs varies with 4 to 16, 26 to 627, 78 to 912, 204 to 2494, and 7.6 to 80.4 respectively. The experimental results of the work reveals the IP-based technique proposed found as many faults as, or more faults than existing techniques in most cases.

A set of tools have been proposed[16] to record and automate test run execution for database application regression tests by Haftmann et al., 2006. Alternative scheduling strategies, parallel Optimistic++, parallel Slice and parallel MaxWeightedDiff heuristics, have been presented to resolve which test runs are to be executed on which machine/thread and in which order. A methodology also been proposed to assess database application testing frameworks. The strategies have been evaluated experimentally in an Linux environment. In all experiments, the global scheduler and the conflict database were installed on one dedicated machine; other five machines each were installed with a local scheduler, a synthetic database application, and a backend relational database for the synthetic database application. IBM DB2 databases were used as backend databases. The synthetic test runs characteristics; Number of test runs, Length of test runs, Number of conflicts, Conflict distribution has been portrayed as 1000. 0 minute to 3 minutes, 1000 to 10000, and uniform distribution respectively. The running time and the number of resets of the parallel Optimistic++, parallel Slice and parallel MaxWeightedDiff heuristics strategies have been studied. The simulated experimental result revealed that the three proposed strategies, parallel Optimistic++, parallel Slice and parallelMaxWeightedDiff achieved linear speed-up in test run execution time.

V.CONCLUSION

Fuzzy based, Genetic based, Greedy based, nongreedy based, Ant colony based approaches/algorithms/frameworks have been implemented in regression testing for the test case selection, minimization, and prioritization process. The implemented approaches focused on time and cost effective strategies, average percentage of fault detection, earlier fault detection, block coverage, and test case sequence. Various metrics such as Irreplaceabilty, EIreplaceabilty, Modified APBC, Additional Modified Lines Of Code Coverage etc have also been derived and evaluated comparatively with existing approaches, and were found effective. Most of the works being focused on single objective and single criterion, certain works focused on muti-objective and multi-criterion strategies and proved to be effective. Algorithms like Modified Cost-Cognizant Test Case Prioritization have been proved more efficient than traditional Genetic Algorithms. Few works have been found focused on dependency structures and found effective in terms of APFD. Historical information based approaches have also been proposed in certain works and found effective in terms of APFD. Risk based approaches has also been found effective in APFD and earlier fault detection.

VI.REFERENCES

[1] C. T. Lin, K. W. Tang, and G. M. Kapfhammer, “Test suite reduction methods that decrease regression testing costs by identifying irreplaceable tests,” *Inf. Softw. Technol.*, vol. 56, no. 10, pp. 1322–1344, 2014.

[2] A. Schwartz and H. Do, “Cost-effective Regression Testing through Adaptive Test Prioritization Strategies,” *J. Syst. Softw.*, vol. 115, pp. 61–81, 2016.

[3] C. Hettiarachchi, H. Do, and B. Choi, “Risk-based test case prioritization using a fuzzy expert system,” *Inf. Softw. Technol.*, vol. 69, pp. 1–15, 2016.

[4] A. Panichella, R. Oliveto, M. Di Penta, and A. De Lucia, “Improving Multi-Objective Test Case Selection by Injecting Diversity in Genetic Algorithms,” vol. 41, no. 4, pp. 358–383, 2013.

[5] C. R. Panigrahi and R. Mall, “A heuristic-based regression test case prioritization approach for object-oriented programs,” *Innov. Syst. Softw. Eng.*, vol. 10, no. 3, pp. 155–163, 2014.

[6] R. Malhotra and D. Tiwari, “Development of a framework for test case prioritization using genetic algorithm,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 38, no. 3, p. 1, 2013.

[7] W. Sun, Z. Gao, W. Yang, C. Fang, and Z. Chen, “Multi-objective test case prioritization for GUI applications,” *Proc. ACM Symp. Appl. Comput.*, pp. 1074–1079, 2013.

[8] S. Kim and J. Baik, “An effective fault aware test case prioritization by incorporating a fault localization technique,” *Proc. 2010 ACM-IEEE Int. Symp. Empir. Softw. Eng. Meas. - ESEM ’10*, p. 1, 2010.

[9] B. Suri and S. Singhal, “Analyzing test case selection & prioritization using ACO,” *ACM SIGSOFT Softw. Eng. Notes*, vol. 36, no. 6,

p. 1, 2011.
[10] C. P. Indumathi and K. Selvamani, "Test Cases Prioritization Using Open Dependency Structure Algorithm," *Procedia Comput. Sci.*, vol. 48, no. Iccc, pp. 250–255, 2015.
[11] E. Rogstad, L. Briand, and R. Torkar, "Test case selection for black-box regression testing of database applications," *Inf. Softw. Technol.*, vol. 55, no. 10, pp. 1781–1795, 2013.
[12] Y. C. Huang, K. L. Peng, and C. Y. Huang, "A history-based cost-cognizant test case prioritization technique in regression testing," *J. Syst. Softw.*, vol. 85, no. 3, pp. 626–637, 2012.
[13] Z. Xu, K. Gao, T. M. Khoshgoftaar, and N. Seliya, "System regression test planning with a fuzzy expert system," *Inf. Sci. (Ny).*, vol. 259, pp. 532–543, 2014.
[14] S. Haidry and T. Miller, "Using Dependency Structures for Prioritization of Functional Test Suites," vol. 39, no. 2, pp. 258–275, 2013.
[15] S. Mirarab, S. Akhlaghi, and L. Tahvildari, "Size-constrained regression test case selection using multicriteria optimization," *IEEE Trans. Softw. Eng.*, vol. 38, no. 4, pp. 936–956, 2012.
[16] F. Haftmann, D. Kossmann, and E. Lo, "A framework for efficient regression tests on database applications," *VLDB J.*, vol. 16, no. 1, pp. 145–164, 2007.