2017 8th International Conference on Information Technology (ICIT)

A Survey on Prioritization Regression Testing Test Case

Dima Suleiman
Computer Science Department
King Hussein Faculty of Computing Sciences, Princess Sumaya University for Technology, Teacher at the University of Jordan
Amman, Jordan
dimah_1999@yahoo.com

Marwah Alian
Basic sciences Department
Faculty of science
Hashemite University
Zarqa, Jordan
Marwah2001@yahoo.com

Prof. Amjad Hudaib
Department of Computer Information Systems, King Abdullah II for Information Technology
the University of Jordan
Amman, Jordan
Ahudaib@ju.edu.jo

Abstract— **Regression testing is a process used to measure the validity of the system during software maintenance. Regression testing process is very expensive and must be introduced each time a modification occurs in software to ensure that the system still work and that the new modification doesn't cause any bugs, this process depends on selecting test cases from a test suite. Selection of test cases is very critical since it affect the regression testing time and effort, so that many algorithms exist to enhance regression testing process. One of the methods used to make enhancements is to select test cases using prioritization testing techniques. Prioritization techniques find the bugs early to improve regression testing efficiency by prioritizing the test cases. In this paper many regression testing prioritization techniques were reviewed and analyzed.**

Keywords—component; Prioritization; Requirements; Regression, Test Cases; Test Suite; Coverage; Model; GA.

I. INTRODUCTION

The testing phase of software development is very expensive; it is used to make sure that the system meets the customer requirements and specifications [1]. Regression testing is used to make ensure that any changes made on the software don't affect the correctness of existing software [2], it's considered as retesting after changes mechanism [3]. Regression testing depends on selecting test cases from a test suite, as a system to be tested is large and complex the test suite will also be like that, so that the selection of suitable test cases will be a very complex process that will affect the efficiency of regression technique [4].

Regression testing initially is used to test the modified parts of the program and then it will retest the overall system [5]. Two main activities used in regression testing selection process: the first one is identifying the unmodified parts affected by the modifications, and the second is the selection of test cases from a test suite [6][7].

Prioritizing the selection of test cases from a test suite will optimize the regression process [6][8], instead of testing all test cases which will waste a time and cost [9], prioritization will schedule the execution of test cases. Many prioritization and selection methods were studied in order to find which methods will increase the efficiency and performance of testing.

The main purpose of test case prioritization is to make scheduling for test cases in order to execute test cases that have highest priority, according to some circumstances, earlier than those with lower priority [10]. The order of selecting the test cases in test suite can be determined according to modified software components; research [11] uses particle swarm optimization (PSO) algorithm to make such an order. PSO algorithm determines the best position of the object so that the test cases with highest priority will be tested first.

Regression testing sometimes made for fixed amount of time so that in this case there is a time constraint [12]. Research shows that regression testing can be under additional conditions related to regression such as time. It depends on previous knowledge of time allotted to perform test cases. Genetic algorithms can be [13] used to make a prioritization testing under time constraints more efficient. Prioritization can be determined by taking time budget into consideration, other choice is to take to consider time budget and fault information in order to increase the average number of faults detected per minute.

The goal of regression testing prioritization techniques is to order test cases in a way reflecting their importance according to specific criteria such as reduced test time or high fault detection rate [14][43].

978-1-5090-6332-1/17/$31.00 ©2017 IEEE

The rest of the paper is organized as follows: Section II provides a review of others work in the same field. Section III describes the prioritization methodology, and conclusion and future work are presented in section IV.

II. Related work

Regression testing is considered as the most expensive phase in software testing. Therefore, regression testing prioritization selects test cases from test suites according to certain priority to be able to save cost and time. Several techniques are used to deal with the problem of regression testing prioritization.

D. Marijan et al. propose a technique called (ROCKET) with tool implementation used to order regression testing test cases. They depend on historical failure data, testing time constraints and domain-specific heuristics. They provide a case study in continuous regression testing of software related to video conferencing. The objective function aim to choose test cases that have highest failure rate and then to maximize test cases that have time constraints [15]. In order to determine the priority of test cases a weighted function is used. The tests with higher weight are those that uncover regression faults in certain iteration of software testing, then the weight is reduced based on detection of faults. The results of their study state that using ROCKET in prioritization will increase the rate of fault detection and provide enhanced fault detection if compared with manually prioritized test cases [14].

Some of prioritization methods are memory less, so that they don't have historical cases to make a comparison with, in other studies the historical effect of test cases are taken into consideration to determine the priority of test cases in detecting the faults. Research [16] uses two benchmark; Siemens suite and Space program and presented new equation. The new equation considers the resource constraint and environment time and depends on three factors: fault detection historical effectiveness, execution history of test cases and finally the last priority of test cases [17].

Bryce and Colbourn suggest an algorithm for regenerating prioritized test suites. Their approach starts by providing each value of each factor with a set of weights then it generates test suites depending on the defined weights. The resulted test suites are biased covering array which is a special kind of a covering array [18]. Another way to make prioritization is to reorder an existing CIT test suites. In this approach concrete test cases can be reused and artifacts from prior versions.

An empirical study that is done on three software subjects is proposed by X. Qu et al. First subject applying both regeneration and pure prioritization of 2-way and 3-way CIT and comparing their effectiveness. The second one is examining many different ways for prioritization control and the third subject uses branch coverage from prior releases to weight interactions. The results of the study illustrate that the 2-way prioritized test suite can detect faults less efficiently than their respective 3-way CIT test suites. These 3 way CIT test suites use interactions weighting that is based on branch coverage to make prioritization, also the results show that regeneration is more efficient at 3-way, but at the start their growth rate are slow. This introduces a suggestion that it should be used only if time allows. However, when there is no code coverage information, 2-way specification may be the best choice [7][19].

In order to improve the regression testing effectiveness such as, test case selection, test case prioritization, retest all, retest all implies re-execution of all available test suites and test case reduction, regression testing methodologies are used which will increase the efficiency.

When a system changed, the test suite will contain two categories of test cases; the original and modified test cases. Regression test prioritization makes a separation between the modified test suite and optimal test cases [20].

Multi-criteria optimization technique is used to generate optimal regression test cases but it is applied in a non-parallelized environment. N. Elanthiraiyan et al. select and prioritize test case suite that is based on a certain objective function by using ant colony optimization methods, using of ant colony will result in increasing the fault coverage with minimal time using Hadoop Map reduce framework. [20]

Accessing to case study applications and their test suites will help in evaluating and implementing of greedy and search based test prioritizers. Zachary and Gregory proposed two types of generated test suites that support the process of prioritization evaluation, in this case synthetic test suites gives more control over test case characteristics [21]. In this research a comparison between greedy and search based prioritization are made.

Their empirical study shows that greedy test suite prioritizer exhibits more overhead time than the hill climbing algorithm. Also their results suggest that hill climbing search-based methods may be used in moderate size suites to make prioritization to cover certain requirements [22].

Many algorithms have been introduced by researchers for test case prioritization such as Metaheuristic search and Greedy Algorithms. A simple greedy algorithm produces a less efficient solution for test case prioritization since it gives a solution that is local optimum. A greedy algorithm uses certain criteria to selects element with maximum weight, the criteria can be branch coverage, function coverage statement coverage, and fault coverage. Metaheuristic search technique such as, Hill climbing and Genetic algorithms generate a better solution to the problem of test case prioritization.

Genetic algorithms use an exact mathematical fitness value for test cases that is going to be prioritized. A global optimal solution can be achieved by genetic algorithm which evaluates individuals from the population of test cases then using mutation, crossover and selection to offer a new generation. This process repeated until a satisfied termination condition is met. N.Sharma et al. compare metaheuristic genetic algorithm with other algorithms in order to prove the efficiency of genetic algorithm over others [23]. Also the Genetic Software Engineering (GSE) method can be used to select a set of test cases; this model functional requirement is called Behavior Trees (BT) [42].

Combinatorial interaction testing (CIT) is classified as black-box testing method; it is used to construct an effective interaction test suite and to identify the faults that are results from the interactions among parameters. After generation interaction test suites that are done by CIT, test case order has to be determined using prioritization techniques. Random prioritization (RP) of test cases is ineffective but on the other hand it is simple.
An alternative and promising candidate is the adaptive random prioritization (ARP) of test cases.

R. Huang et al. propose ARP strategy as an enhancement of RP strategy, in order to prioritize interaction test suites using interaction coverage information without requiring source-code related information. In this research they make simulation studies which show that the proposed ARP has better performance than RP strategy. Also the results show that ARP can be more time saving than ICBP and it maintains similar effectiveness. Techniques for prioritization include statement coverage, function coverage and fault finding exposure, or interaction coverage.

Prioritization can either be code-based or model based test cases [24]. In code based the test cases prioritized using the source code of the system, most of prioritizing techniques is based on code. On the other hand, system models are used in model based test cases, this type of techniques used to improve the early detection of faults. Model based test case is better than code based test cases since it is inexpensive and may be sensitive to the correct or incorrect information retrieved from testers or developers [24].
In this paper we provide analysis for the existing test case prioritization methods that is related to requirement, coverage and model based.

III. PRIORITIZATION METHODOLOGY

There are many prioritization techniques have been proposed. In this survey we provide a review and classification for the existing test case prioritization techniques. We classify those techniques into requirement based, coverage based and model based. Those classes are shown in Fig. 1. As shown in the figure we consider coverage based class to be related to fault finding interaction, total coverage using GA code coverage, and multiple control flow techniques. Also, requirement class is considered to be related to customer requirement, requirement complexity, and volatility of requirement fault proneness prioritization techniques. More details about these classes are given in the following subsections.

1) *Requirements based test cases Prioritization:*

Quality of software related to how the software meets the customer requirements. In order to be able to -improve the software quality, the test cases designer must design test cases to cover all the customer requirements which require much time and effort. Analytic Hierarchy Process AHP [25] used to prioritize test cases; this process is related to the goal of decision making.

The test cases are retrieved and stored according to requirements changes. After retrieving the test cases a processing is performed to determine the effect of requirement changes on the test cases stored in a database, test cases will be ranked according to the changes and then test planning can be performed in the results [25]. The main factories for prioritizing test cases to meet customer requirements are: Customer Assigned Priority according to the importance of requirements (CP), Volatility of Requirements which related to the number of changes in requirements (RV), Implementation Complexity (IC) of the requirements according to development team opinion and finally the Fault Proneness (FP): giving a score of each requirement by considering previously faulty requirements that may have the same defect again.

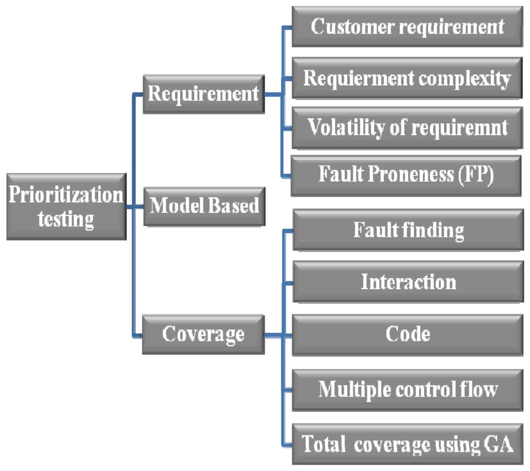

Fig.1: prioritization methodologies

In research [25], AHP is used to rank the test cases and the main focus of research is to store and retrieve the information , the process separated into set of steps:

A. Storage of documents relating to software testing:

There are two parts of the storage method: indexing test case and determining the weights of each. Index creation is achieved by creating a set of words to represent the document, this can be done by taking the data in test cases and separating them into terms, and then removing terms without meaning and as a last step make word stemming by removing suffixes to return the word to its root. The weight of the indexes can be used to determine the value of index. After applying the two parts the file structure will be used to store indexes.

B. Test case Retrieval:

System testers create a query of required test case which affected by the changes in requirements. After that the testers will make comparisons between the indexes of each test case from a query and the ones stored in a database. Similarity value will be used to determine highest values test cases.

C. Ranking test cases:

AHP process will be applied to assess ranking using the following steps:

(1) Determining the affected test cases by the requirements changes.
(2) Affected test cases are scored.
(3) Comparing test cases according to the importance.
(4) Calculating test case weight.
(5)Analyzing the results of requirements changes, that has sequential effects.

D. Evaluating the proposed method:

The accuracy of results is assessed using a metric either by measuring the effectiveness of test case retrieval or by measuring the prioritizing test cases effectiveness [25].

In [9] contribution is in the requirement based system for the schema of prioritizing test case levels. Their objective is to enhance user satisfaction using software that has the quality and cost efficiency. Also, they aim to have improvement in the rate of fault detection. The proposed model prioritizes test cases based on a number of factors, such as Implementation Complexity, Customer priority, Application Flow, Usability, Requirement changes and Fault Impact. Their proposed technique seems to improve the rate of fault detection [9]

Raju and Uma [26] introduce requirement based system for test case prioritization scheme using Genetic Algorithm; their objective is to check for more severe faults in earlier phases and to enhance software quality. They propose a set of prioritization factors which may be concrete or abstract in order to design their system which make prioritization for the test cases based on six factors: customer priority, the complexity of implementation, the completeness, and changes in requirements, fault effect and traceability. The evaluation of these priorities is measured using APFD and PTR evaluation metrics.

In the proposed technique [26], two types of applications have been involved. The proposed method gives more improvement in the rate of severe faults detection and the number of test cases run to check and find the injected fault is less in case of the new technique. They show that their method prioritizes the test cases in the involved projects in an efficient way based on the factors and weights that are given to test cases and this reduce the cost of execution for the project.

2) Coverage based test cases Prioritization:

One of criteria that regression testing is based on is coverage. Coverage has many types such as fault coverage, branch coverage, interaction coverage, code, multiple control flow coverage and total coverage using GA [27][28]. Code coverage can be used as quantitative measurement of source code which can be expresses as ratio of number of lines in a source code [29].

Many methods use code coverage such as statements total prioritizations which prioritize test cases according to the total number of statements they cover, loop total prioritization that depends on number of loops, branch total prioritization measured by the number of executed branches and condition total prioritization that depends on number of boolean expression executed [3][30].

A new prioritization technique [31] takes into consideration the coverage and the historical fault information by introduction fault localization methods. It maintains the test cases that have high coverage and assign high priority for them. The fault localization methods facilitate the debugging technique since it considers where the faults were exists after isolating the faults location. [31] Introduce a new method name it as Fault Aware Test Case Prioritization (FATCP)

FATCP consider both the fault localization methods and coverage-based test case prioritization. This approach may have additional cost which can be reduced by implementing an automation tool. This tool is the Average Percentage of Faults Detected Parsing Tool (APT). Fault localization main idea is that when executing a program with many failed test cases it will have many faults. FATCP has some limitations related to that the removal of the faults in the initial program affect the ability of fault detection. In addition to that the new approach highly relies on TRANTULA performance, also if the estimation of the fault location is wrong the reliability cannot be guaranteed.

Code coverage Árpád et al. experimentally applies change-based test selection technique to the web browser engine project WebKit which is an open source. They aim to encounter the technical difficulties and to find the benefits that are expected if this technique is introduced in the real build process. They adapted this method to be used in this real and large open source project. Their adapted change-based test selection method has been successfully implemented in the live environment of the project and can be used in detecting high rate of failures [32][33].

There are many prioritization criteria that have been used in order to make improvement in the detecting rate of faults. One recent criterion that is used in prioritization is that of coverage interaction. In many studies the criteria has been useful, but these studies do not consider the cost of tests when prioritizing test cases. However, Rene´e et al. propose combinatorial interaction coverage metric depending on cost, they implement an algorithm to make computation for the new metric and they apply an empirical study in which they involve three subject web applications. They examine the new criteria in order to prioritize test cases by interaction coverage that takes the costs of test cases into consideration. The results about two studies imply that using the new metric in prioritization improves the detection rate of faults with relation to cost while an interesting result provided by the third study, that is the distribution of t-tuples in the selected test cases which influence the success of the cost-based metric [34].

In [35] present an automating technique for test case prioritization using genetic algorithm and they utilize Multi-Criteria Fitness function. They use multiple metrics for control flow coverage where coverage degree for multiple conditions is checked and measured for statements that the test case covers using these metrics then a weight is given to the metrics by the number of faults revealed and how they sever.

Patipat and Lachana [36] introduce a new algorithm to be used in prioritizing test cases depending on total coverage, they use genetic algorithm that is modified then they compare the performance of this algorithm depending on the average percentage of conditions that are covered. They compare the execution time with five algorithms and they show that their technique achieves 100% code coverage.

Manika and Sona [37] proposed a 3-phase technique in order to solve the problem of test case prioritization. In starting phase, redundant test cases are removed while in the second phase, a minimal set of test cases that covers all faults and with minimum execution time are selected from the test suite. In this phase they use the algorithm of multi objective particle swarm optimization (MOPSO) which role is to improve and enhance fault coverage and execution time. In the third phase, priority is allocated to test cases that obtained from the second phase, which is computed as the ratio of fault coverage and time for execution. Test cases that have high ratio value will have higher priority. Unselected test cases in phase 2 are added sequentially to the test suite. Their analysis depends on the maximum fault coverage and the minimum time for execution where the proposed technique is compared with different cases of ordering such that no ordering, random ordering and reverse ordering technique. The Average Percentage of fault detection for the proposed approach outperformed all techniques compared with [37].

[38] Proposes a set of prioritization algorithms that belongs to coverage-related technique, these algorithms are depending on historical test factor effect on assigning priority for test cases. They argue that there is more remaining errors in the function that changed than the unchanged one. Thus, the priority can be assigned for test cases depending on counting changed points, which is covered by the call path of a function. In their work they start with same value for all test cases priority, then all function call paths are navigated to compute how many change point in any path. By utilizing matrix, the number of change points is checked, those points are covered by each test case and the obtained number is considered as the test case priority. Finally, test cases will be arranged according to their priority. The three algorithms are; prioritizing by Numbers of Changes (PNC), prioritizing by Times of Changes occurrence (PTC), MIX which is integration between PTC and PNC. If there are more changes in the software, algorithm PNC is used while if the software changes are concentrated in a small number of functions, algorithm PTC has more efficiency and through regression testing process, two algorithms PNC and PTC can be combined together.

3) *Model Based test cases Prioritization:*

Different types of test case prioritization algorithms have been exist such as code-based test case prioritization and model-based test case prioritization. However, in code-based test case prioritization, source code is the main factor in the prioritization process. While a system's model is involved to prioritize test cases in Model-based. System models have been utilized to get some aspects of how the system behaves. Model-based test prioritization has lower cost than code-based test prioritization since the model is executed faster. But they are sensitive to the correctness factor of information that is given by the testers or developers. [39]

[39] presents an experimental study in order to evaluate code-based and model-based test prioritization methods they depend on the effectiveness of how earlier fault can be detected in the updated and changed system. In their work they select statement coverage as code based prioritization and call it Heuristic #1. In the model-based test prioritization they use another technique which is Heuristic #2. It depends on marking transitions. When modifying source code, developers define model elements that are relative to those changes, the transitions are called marked transitions.
By experiment it is demonstrated that model based test prioritization can enhance the early fault detection more than code based test prioritization because it is executed faster than the execution of the actual system [40].

Harish and Naresh propose a module-coupling effect based test case prioritization method. This method is based on the coupling information among the modules of the program such that checking the type of coupling between modules and if it is known, then the priority is assigned and developed based on this coupling such that the modules that have bad type of coupling is given a higher priority over test cases in other modules. It has been proved that this method can check the critical modules that have higher chance of including critical faults; they evaluate their approach using a case study of software having ten modules [41].

Sanjukta et al. [Mohanty S. et al. 2011] propose a new technique that prioritizes the test cases in order to start regression testing for Component Based Software System (CBSS). In the beginning, components and their states that are changed depending on software systems are clarified using UML diagrams. After that, they are implemented using Component Interaction Graph (CIG) which describes the interrelations between components. Then this graph is used as input to the proposed prioritization algorithm using old test cases and the output is a prioritized test suit. The proposed algorithm takes into consideration the number of state changes and the number of database access that happened through the state changes by test case. This algorithm effectively maximizes the objective function and minimizes the cost of regression testing when applied to few number of JAVA projects. A comparison between the techniques can be show in Table I.

Different types of test case prioritization algorithms have been reviewed in this research, they are classified into requirement, coverage and model based. It is considered that model based algorithms are faster than algorithms in other classifications since it uses models rather than source code in testing process. Selecting a prioritization algorithm depends on cost, earlier fault detection and priority criteria that can be an important factor to make improvement in the detecting rate of faults.

IV. Conclusion and Future work

The objective of test case prioritization methods is to enhance the effectiveness of regression testing and to detect faults as early as possible. This can be achieved by ordering the test cases such that the test case with most benefit executed first and given with higher priority. In our study of existing prioritization technique we focused on three main categories: requirement, coverage and model based techniques.

Several Techniques were used to prioritize test cases, some of them used GA and others used multi objective swarm optimization, some depended on historical test cases while others are memory less. In order to determine which technique to use it is important to understand the requirements of the user in this case according to user requirements you can determine the best technique in requirement category. On the other hand you can choose either to use model or code based test cases were most of methods used code based test cases. When choosing code based test model then you can choose code coverage and from this category you have function, statement, loop, branch and many other type. In future many criteria may be included like code coverage related to databases and distributed databases and how to extract test cases form them more efficiently, also other evolution methods can be used in addition to using GA, such as Differential evolution (DE) algorithms.

TABLE I: TEST SUITE PRIORITIZATION TECHNIQUES

Year	**Title**	**Author**	**Optimization Type**	**Used Algorithm Metric model**
2008	Incorporating varying requirement priorities and costs in test case prioritization for new and regression testing,	K. Ramasamy, S.A. Mary,	requirement	metric
2009	Experimental Comparison of Code Based and Model model Based Test prioritization,"	Korel et al	Model Based	Heuristic #1 Heuristic #2
2010	Using Synthetic Test Suites to Empirically Compare Search-Based and Greedy Prioritizers	Zachary D. Williams, Gregory M. Kapfhammer	requirement	greedy search-based HillCimbing
2010	An Effective Fault Aware Test Case Prioritization by Incorporating a Fault Localization Technique	Sejun Kim, Jongmoon Baik	Coverage	Fault Localization
2011	Requirement-Based Test Case Generation and Prioritization	Yasmine Ibrahim Salem, Riham Hassan	requirement	Genetic Software Engineering Behavior Trees
2011	A MODEL BASED PRIORITIZATION TECHNIQUE FOR COMPONENT BASED SOFTWARE RETESTING USING UML STATE CHART DIAGRAM	Sanjukta Mohanty, Arup Abhinna Acharya, Durga Prasad Mohapatra,	model based	GIC dependant algo
2011	Test suite prioritization by cost-based combinatorial interaction coverage",	Rene´e C. Bryce, Sreedevi Sampath, Jan B. Pedersen, Schuyler Manchester,"	interaction coverage	metric
2012	Factors Oriented Test Case Prioritization Technique in Regression Testing using Genetic Algorithm , ARTICLE in EUROPEAN JOURNAL OF SCIENTIFIC RESEARCH	Dr Raju Shanmugam , G. V. Uma,	Requirement	GA
2012	Test Case Prioritization for Regression Testing Based on Function Call Path,	ZHANG Zhi-hua MU Yong-min TIAN Ying-ai ,	coverage-related technique	PTC , PNC, MIX
2012	Code Coverage-Based Regression Test Selection and Prioritization in WebKit"	Árpád Beszédes, Tamás Gergely, Lajos Schrettner, Judit Jász, László Langó, Tibor Gyimóthy,	Code coverage	Adapt change-based test selection method for WebKit
2012	SOFTWARE TESTING SUITE PRIORITIZATION USING MULTICRITERIA FITNESS FUNCTION	Amr AbdelFatah Ahmed,Dr. Mohamed Shaheen,Dr. Essam Kosba	multiple control flow coverage	GA
2012	On the Fault-Detection Capabilities of Adaptive Random Test Case Prioritization: Case Studies with Large Test Suites	Zhi Quan Zhou, Arnaldo Sinaga and Willy Susilo	code coverage	Jaccard Distance (JD) and Coverage Manhattan Distance (CMD)
2012	Test Case Priorization Incorporating Ordered Sequence, of Program Elements	Kun Wu, Chunrong Fang, Zhenyu Chen, Zhihong Zhaio	coverage	ART
2014	Importance of Selecting Test Cases for Regression Testing	Kamna Solanki , Dr. Yudhvir Singh	coverage	metric

2014	"Test Case Prioritization using Multi Objective Particle Swarm Optimizer"	Manika Tyag, Sona Malhotra	fault coverage	multi objective particle swarm optimization (MOPSO)
2014	Optimized Test Case Prioritization with multi criteria for Regression Testing	KanwalpreetKaur, Satwinder Singh	coverage	metric
2015	Test Cases Prioritization for Software Regression Testing Using Analytic Hierarchy Process	Piyakarn Klindee, Nakornthip Prompoon	Requirement	Analytic Hierarchy Process(AHP)
2015	Total Coverage Based Regression Test Case Prioritization using Genetic Algorithm	Patipat Konsaard, Lachana Ramingwong	total coverage code coverage	GA
2015	A Coupling Effect Based Test Case Prioritization Technique	Harish Kumar, Naresh Chauhan	Module coupling	metric

REFERENCES

[1] Salem Y., Hassan R., Requirement-Based Test Case Generation and Prioritization, 978-1-61284-185-4/111$26.00 ©2011 IEEE

[2] Presitha Aarthi. M, Nandini. V (2015), A Survey on Test Case Selection and Prioritization, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 5, Issue 1, January 2015.

[3] Karambir,Rani R.(2013), Prioritize Test Case Survey for Component-Based Software Testing, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 6, June 2013.

[4] Kavitha R. , Sureshkumar N.(2010) , Test Case Prioritization for Regression Testing based on Severity of Fault, R.Kavitha et. al. / (IJCSE) International Journal on Computer Science and Engineering, Vol. 02, No. 05, 2010, 1462-1466

[5] Mohanty S., Acharya A., Mohapatra D.(2011), A SURVEY ON MODEL BASED TEST CASE PRIORITIZATION, International Journal of Computer Science and Information Technologies, Vol. 2 (3) , 2011, 1042- 1047 1042

[6] Zachary D. Williams, Gregory M. Kapfhammer(2010), Using Synthetic Test Suites to Empirically Compare Search-Based and Greedy Prioritizers,GECCO'10, July 7.11, 2010, Portland, Oregon, USA. Copyright 2010 ACM 978-1-4503-0073-5/10/07 ...$10.00.

[7] Xiao Q.,Cohen M.(2013), A Study in Prioritization for Higher Strength Combinatorial Testing, 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops.

[8] Yoo S., Harma M,(2007) Regression Testing Minimisation, Selection And prioritisation : A Survey, SOFTWARE TESTING, VERIFICATION AND RELIABILITY Softw. Test. Verif. Reliab. 2007; 00:1–7 (DOI: 10.1002/000).

[9] Ramasamy K.(2008),Incorporating varying Requirement Priorities and Costs in Test Case Prioritization for New and Regression testing, Proceedings ofthe 2008 International Conference on Computing, Communication and Networking (ICCCN 2008) 978-1-4244-3595-1/08/$25.00 \02008 IEEE.

[10] Huang R., Chen, Li Z.,WangYansheng R.(2014)," Adaptive Random Prioritization for Interaction Test Suites",Proceedings of the 29th Annual
ACM Symposium on Applied Computing, pp 1058-1063, SAC'14 ACM 2014.

[11] Khin H., YoungSik C. , Jong P.(2008), Applying Particle Swarm Optimization to Prioritizing Test Cases for Embedded Real Time Software Retesting, IEEE 8th International Conference on Computer And Information Technology -Workshops 978-0-7695-3242-4/08 $25.00 © 2008 IEEE DOI 10.1109/CIT.2008.Workshops.104.

[12] Walcott K. R., Lou Soffa M., Kapfhammer G. M., Roos R. (2006), imeAware Test Suite Prioritization, ISSTA'06, July 17–20, 2006, Portland, Maine, USA. Copyright 2006 ACM 1595932631/ 06/0007.

[13] Seema S.(2012), A Genetic Algorithm for Regression Test Sequence Optimization, International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 7, September 2012.

[14] Marijan D., Gotlieb A.,Sen S.(2013), "Test Case Prioritization for Continuous Regression Testing: An Industrial Case Study" 2013 IEEE International Conference on Software Maintenance.

[15] Xiaofang Z. Changhai B. ,Xu Q.(2007), Test Case Prioritization based On Varying Testing Requirement Priorities and Test Case Costs, Seventh Inernational Conference on Quality Software (QSIC 2007) 0-7695-3035- 4/07 $25.00 © 2007 IEEE.

[16] Fazlalizadeh Y., Khalilian A., Abdollahi Azgomi M. and Parsa S.(2009), Prioritizing Test Cases for Resource Constraint Environments Using Historical Test Case Performance Data, 978-1-4244-4520-2/09/$25.00 ©2009 IEEE.

[17] Gupta R. and Soffa M.(1995) , Priority Based Data Flow Testing, 1063-6773/95 1995 IEEE.

[18] Bryce R. and Colbourn C.(2006), "Prioritized interaction testing for pair-wise coverage with seeding and constraints," Journal of Information and Software Technology, vol. 48, no. 10, pp.960–970, 2006.

[19] Suri B, Nayyar P.(2011), COVERAGE BASED TEST SUITE AUGMENTATION TECHNIQUES-A SURVEY, International Journal of Advances in Engineering & Technology, May 2011.

[20] Elanthiraiyan N.,Arumugam C.(2014),"Parallelized ACO Algorithm for Regression Testing Prioritization in Hadoop Framework",Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference. IEEE.

[21] Dobuneh M., Jawawi D., Malakooti M.(2013), Web Application Regression Testing: A Session Based Test Case Prioritization Approach, ISBN: 978-0-9891305-1-6 ©2013 SDIWC.

[22] Williams Z., Kapfhammer G, Using Synthetic Test Suites to Empirically Compare Search-Based and Greedy Prioritizers .

[23] Sharma N., Sujata, N. Purohit (2014),Test Case Prioritization Techniques An Empirical Study", High Performance Computing and Applications (ICHPCA), 2014 International Conference, IEEE.

[24] Mohanty S., Acharya A.,Mohapatra D.(2011), A MODEL BASED PRIORITIZATION TECHNIQUE FOR COMPONENT, 978-1-4244-8679-3/11 ©2011 IEEE.

[25] Klindee P. ,Prompoon K. , Test Cases Prioritization for Software Regression Testing Using Analytic Hierarchy Process, 2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE).

[26] Uma R.(2012), Factors Oriented Test Case Prioritization Technique in Regression Testing using Genetic Algorithm, EUROPEAN JOURNAL OF SCIENTIFIC RESEARCH · APRIL 2012.

[27] Kaur K. ,Singh S.(2014), Optimized Test Case Prioritization with multi criteria for Regression Testing, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 3, Issue 4, April 2014.
[28] Wu K., Fang C., Chen Z., Zhaio Z(2012), Test Case Priorization Incorporating Ordered Sequence, of Program Elements. 2012 IEEE.
[29] Solanki , Singh Y(2014), Importance of Selecting Test Cases for Regression Testing, IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 4, Ver. IV (Jul – Aug. 2014).
[30] Zhou Z., Sinaga A. and Susilo W.(2012), On the Fault-Detection Capabilities of Adaptive Random Test Case Prioritization: Case Studies with Large Test Suites, 2012 45th Hawaii International Conference on System Sciences
[31] Kim S., Baik J.(2010), An Effective Fault Aware Test Case Prioritization by Incorporating a Fault Localization Technique, ESEM'10, September 16–17, 2010, Bolzano-Bozen, Italy. Copyright 2010 ACM 978-1-4503-0039-01/10/09…$10.00.
[32] Biswas S. and Mall R.(2011) , Regression Test Selection Techniques: A Survey, Informatica 35 (2011) 289–321.
[33] Beszédes A., Gergely T, Schrettner L., Jász J, Langó L and, Gyimóthy T(2012), Code Coverage-Based Regression Test Selection and Prioritization in WebKit, 978-1-4673-2312-3/12/$31.00 c 2012 IEEE.
[34] Rene´e C. Bryce, Jan B. Pedersen(2011), Test suite prioritization by cost-based combinatorial interaction coverage , Int J Syst Assur Eng Manag (Apr-June 2011) 2(2):126–134 DOI 10.1007/s13198-011-0067-4.
[35] Ahmed A., Shaheen M. ,Kosba K.(2012) , SOFTWARE TESTING SUITE PRIORITIZATION USING MULTICRITERIA FITNESS FUNCTION, ICCTA 2012, 13-15 October 2012, Alexandria, Egypt.
[36] Konsaard P., Ramingwong L.(2015) ,Total Coverage Based Regression Test Case Prioritization using Genetic Algorithm, 978-1-4799-7961-5/15/ ©2015 IEEE
[37] Tyagi M., Malhotra S.(2014), Test Case Prioritization using Multi Objective Particle Swarm Optimizer, 978-1-4799-3140-8/14/ ©2014 IEEE.
[38] Ying-ai Z.(2012) , Test Case Prioritization for Regression Testing Based on Function Call Path, 2012 Fourth International Conference on Computational and Information Sciences.
[39] Korel B., Koutsogiannakis G.(2009), "Experimental Comparsion of Code Based and Model model Based Test prioritization," IEEE 2009
[40] Sun F.(2014), Regression Testing Prioritization Based on Model Checking for Safety-Crucial Embedded Systems, 2014 .
[41] Kumar H., Chauhan N.(2015) "A Coupling Effect Based Test Case Prioritization Technique",2015.
[42] Sale Y., Hassan R.(2011), Requirement-Based Test Case Generation and Prioritization, 978-1-61284-185-4/111$26.00 ©2011 IEEE.
[43] Alian, M., Suleiman, D., & Shaout, A. (2016). Test Case Reduction Techniques-Survey. International Journal of Advanced Computer Science & Applications, 1(7), 264-275.