An Evolution of Test Case Prioritization Techniques

Heena Singhal
M.Tech Computer Science and Engg,
Ajay Kumar Garg Engineering College,
Ghaziabad, UP, India

Kirti Tyagi
Department of Computer Science and Engg,
Ajay Kumar Garg Engineering College,
Ghaziabad, UP, India

ABSTRACT
Test case prioritization is the way of arranging the test cases on the basis of same defined criteria so that fault detection may be made as earlier as possible and hence cut down the cost incur during testing process. Due to day by day increasing complexity of the software system, a lot of test cases are required to execute for effective validation and verification that adds to cost and time. Any prioritization technique schedules the test cases in the way that runs the test cases with higher priority before the test cases with lower priority. Present paper gives a comparative overview of various criteria, techniques and methods used for test cases prioritization for the component based software system from the year 1999 to present.

Keywords
Regression testing, software development life cycle, component based software system, test case prioritization.

1. INTRODUCTION
Software testing is the task of evaluating the correctness of the system or its component(s). Testing plays a vital role during software development life cycle (SDLC). It takes almost half the total time of the SDLC. Testing may be handled either manually or automatically. In a general context, automated testing is preferred over the manual testing for big test suite.

Regression testing is done when some modification is made to present software system. It is quite possible whenever a change is made to present software system, the other area within the system may have been affected by this change. Retesting is performed to verify that a fix bug has not resulted into another bug. The altered parts of the system are tested first during the regression testing and later the entire system needs to be retested, so that the confidence regarding the software performance can be built against the modifications introduced and thus ensure that modified software did not introduce new faults into the present system. Since, the size of test suite is very large, system retesting taken away a large amount of time as well as computing resources. Retesting process may last for minutes, hours, days, or even month. The main issue faced by developers during system retesting is the way of scheduling the test cases for execution. Prioritization of test cases tries to combat this issue.

Test case prioritization (TCP) is the process of executing the test cases with the higher priority before the test cases with lower priority on the basis of some fitness value. Retesting is an important phase of software maintenance, but the phase incurs high cost. To compensate with this high cost phase, prioritization of test case is done by software tester so that test case that cost high may execute earlier in the SDLC lifecycle.

The component based software system (CBSS) is the most active part of software engineering. It has grasped the attention of various researchers and developers. It is more generalized approach as compared with existing software engineering approach. The CBSS is one key technique for developing software system. The CBSS may be implemented in many disciplines with many different techniques. The CBSS consists of various components which are developed by third party vendor, however this will reduced the overhead for the developer and also increases the system openness by modify or add a component. It will incur the cost and time, the two important factor of software development life-cycle.

2. TEST CASE PRIORTIZATION
Test case prioritization at first is put forth in regression testing that aims to test the changed software system during software evolution by reusing the test suites of its previous version. TCP gives the way of scheduling the order of test cases so as to maximize fault detection rate at the earlier stage of SDLC. The main performance goal regarding the use of TCP is the rate of fault detection. Test cases must be executed in a way that enhances the rate of fault detection and also it discovers the high risk faults at earlier stage of testing phase of SDLC.

TCP Problem Statement may be summarized as:

Given: a program x and its modified version x'; a test suite t; the set of permutations of test suite xt; a function from xt to the real numbers f.

Problem: Find t ' ∈ xt such that (∀t") (t" ∈ xt)

(t"≠ t') [f (t') ≥ f (t")]

Here, xt represents the set of all possible ordering of prioritizations of t, and f is a function that, applied to any such ordering, yields an award value for that ordering. The definition assumes that higher award values are preferred over the lower ones [2].

There are number of possible goals for TCP such as:

(1) To improve the rate of fault detection.

(2) To increase the code coverage rate.

(3) To enhance the reliability of the system.

The objectives of prioritization of test cases include:

(1) To increase the rate of fault detection i.e. unveiling faults at early stage of testing process.

(2) To increase the rate of detection of high risk faults at early stage of testing process.

(3) To increase the coverage of programmable code in the system under test at a high pace.

(4) To increase the reliability of the system under test at a high pace.

(5) To increase the probability of revealing regression wrength related to version specific code changes at early stage of testing process.

The number of methods has been introduced to prioritize the test case and they are as follows:

(1) Code based TCP

(2) Statement based TCP

(3) Branch based TCP

(4) Function based TCP

(5) Model based TCP

3. LITERATURE REVIEW: AN EVOLUTION OF TEST CASE PRIORITIZATION

In last few decade there were number of publication that discussed the concept of TCP and its related techniques. In this section of paper survey of various prioritization method and related work has been done.

3.1 Year 1999

Test Case Prioritization: An Empirical Study

In [1] Gregg Rothermel et al describes various techniques for TCP and provides with an empirical result that measures the effectiveness of the techniques for enhancing the rate of fault detection in the program. An improved rate of fault detection provides earlier feedback to the system under the regression testing and helps debuggers to correct fault at the earlier stage of testing process. The paper also highlights trade-offs between numbers of prioritization techniques. The nine different techniques of TCP are considered. They are as follows (1) zero prioritization of test cases, (2) random prioritization, (3) optimal prioritization, (4) total branch coverage prioritization, (5) additional branch coverage prioritization, (6) total fault exposing potential prioritization, (7) additional fault exposing potential prioritization, (8) total statement coverage prioritization and (9) additional statement coverage prioritization. The two research oriented question raised are (1) may prioritization improves the rate of fault detection and (2) how various prioritization discussed in the paper are compared to one another in terms of fault detection rate. The result prove that these prioritization techniques significantly improves quick fault detection by the given test suites. Also the same result arrived even for the least sophisticated prioritization techniques.

3.2 Year 2000

Prioritizing Test Cases for Regression Testing

In [2] Sebastian Elbaum et al discussed some of research question i.e. (1) is prioritization be effective while applied at altered versions of programs, (2) what type of trade-offs presents in between different prioritization techniques such as fine granularity prioritization and coarse granularity prioritization and (3) may the involvement of fault proneness measure improves the effectiveness of prioritization techniques. This paper presented with fourteen different types of TCP techniques that are classified under three groups. The first group known as control group comprises of two prioritization technique that serves as experimental control techniques and they are named as (1) random ordering and (2) optimal ordering. The second group known as statement level group comprises of four fine grain techniques and they are named as (3) total statement coverage prioritization (4) additional statement coverage prioritization (5) total fault exposing potential prioritization and (6) additional fault exposing potential prioritization. The third group known as function level group comprises of eight coarse grain prioritization techniques and they are named as (7) total function coverage prioritization (8) additional function coverage prioritization (9) total fault exposing potential function level prioritization (10) additional fault exposing potential function level prioritization (11) total fault index prioritization (12) additional fault index prioritization (13) total fault index with fault exposing potential coverage prioritization and (14) additional fault index with fault exposing potential coverage prioritization. The metric weighted average percentage of fault detected (APFD) is evaluated over the life of the test suites. The value of APFD ranges from 0 to 100, higher the value of APFD means higher the rate of fault detection. The APFD measure ranked the different techniques, the ANOVA analysis was used to find whether the techniques differed from each other while Bonferroni analysis gives the comparison between different techniques i.e. how they are different from each other. Also in this paper threats to validity of experimental study are described. The total of three threats have been described, they are (1) threats to internal validity (2) threats to external validity and (3) threats to construct validity. The result shows the differences exist in between the rates of fault detection among the stated prioritization techniques and also the practical consequences of result were shown.

3.3 Year 2001

3.3.1 Understanding and Measuring the Sources of Variation in the Prioritization of Regression Test Suites

In [3] Sebastian Elbaum et al observed the unknown variance that point out the additional factors that affects the prioritization effectiveness along with the target program and the TCP techniques. In previous work Sebastian Elbaum explained number of prioritization techniques that significantly improves the rate of fault detection. Further he explained that the rate of fault detection is in close proximity of the program under test. The three research question that were raised due to stated observation are (1) whether there are factors other than the prioritization techniques and the tested program, (2) what type of metrics are used to represent each factor and (3) was the inclusion of other factor effect the prioritization techniques. To formulate these questions a series of experiment was conducted that include three factors (1) program structure, (2) test suite composition and (3) change characteristics. During experimentation two types of variable were used dependent variable and independent variable. The dependent variable include the use of APFD measure while independent variable consists of four types of variable and they are (1) subject program, (2) prioritization techniques, (3) version and changes introduce in the program and (4) test suite characteristics. The experiment comprises of eight programs and each program consists of a baseline version and twenty nine version of which each contains multiple faults. For every baseline version, number of test cases exists. The selected prioritization techniques that were described in previous work are further briefly described here and are as follows (1) total function coverage, (2) total statement coverage, (3) additional function coverage, (4) additional statement coverage, (5) total fault index, (6) additional fault index and (7) optimal. To understand the dimensionality of the program, the information presented by the each variable and to predict the correlation among each variable, the principal

component analysis (PCA) has been performed over the collected data and metrics. The PCA discovered the six underlying source of variation. To examine the interrelationship among the dependent and independent variable regression analysis have been used. The regression analysis was performed in two phase. In first phase, regression analysis over individual variable has been evaluated against the APFD measure criteria and in second phase multiple regressions was done to explore the certain factors that affects different prioritization techniques. The result of the study shows that new factors leads to the growth of more powerful TCP techniques, while high prediction capability of regression analysis helps to evaluate the new ordering of test suites.

3.3.2 Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization

In [4] Sebastian Elbaum et al presents a new metric for evaluating the rate of fault detection that includes the varying test cases and fault cost of prioritized test cases. The metric used in previous study did not incorporate the varying test cost and fault severity of prioritized test cases. However, for the uniform test cost and fault detection severity, the metric APFD measures the weighted average cumulative percentage of detection of faults over the life of test suites. The case study has been exercised to show the practical application of the new metric introduce. This study gives rise to some of practical question in applying the new metric to evaluate the prioritization of test cases. The example have been derived that disapprove the use of old APFD measure due to inclusion of varying cost of test and fault severity and hence new cost cognizant metric $APFD_C$ has been invented by adapting the old APFD metric. The graph has been represented for the $APFD_C$ where horizontal axis denote the percentage of total cost of test case incurred while the vertical axis denote the percentage of total severity of fault detection. In the study five different test case cost distribution have been used and they were as follows: (1) unit test case cost, (2) random test case cost, (3) normal test case cost, (3) Mozilla test case cost, where Mozilla is an open source web browser that include large number of developers and testers and (5) QTB test case cost where QTB is a software system that include more than 300KLOC. Also the study described the fault severity detection which three in number and as follows: (1) unit fault severity detection, (2) Mozilla linear fault severity detection, and (3) Mozilla exponential fault severity detection. Thus with the given distribution of test case cost and fault severity detection, a total of fifteen possible combination of the pair. But in the study, attention was restricted to nine combination of test cases cost and fault detection severity.

3.4 Year 2002
Test Case Prioritization: A Family of Empirical Studies

In [5] Sebastian Elbaum et al shows that comparative effectiveness of various prioritization techniques varies significantly over the target program. The total of eighteen prioritization was taken into the accounts that were subdivided into three groups. The first group known as comparator group consists of two techniques (1) random ordering and (2) optimal ordering. The second group, statement group consists of four fine grain techniques (3) total statement coverage, (4) additional statement coverage, (5) total fault exposing potential (FEP) and (6) additional fault exposing potential. The third group, function group consists of twelve coarse grain techniques (7) total function coverage, (8) additional function coverage, (9) total FEP function level, (10) additional function level FEP, (11) total fault index (FI), (12) additional FI, (13) total FI with FEP coverage, (14) additional FI with FEP coverage, (15) total difference (DIF) based, (16) additional DIF based, (17) total DIF with FEP and (18) additional DIF with FEP. The APFD metric is mathematically formulated as:

$APFD = 1 - (TF_1+TF_2+\ldots+TF_m)/nm + 1/2n$, where TF_i is the first test case in new ordering T'.

3.5 Year 2004
Empirical Studies of Test Case Prioritization in a JUnit Testing Environment

In [6] Hyunsook Do et al extended the previous work of TCP to numerous other language paradigm. In early studies concept was bind to C language only, but whether it works for other language was not known. In this paper, TCP is deployed in language Java under JUnit framework. Hyunsook performed controlled experiment to check the TCP effectiveness over Java programs under JUnit testing framework. The result shows significant improvement in rate of fault detection of test suite of JUnit. The paper also presents the differences that exist in between the testing paradigm and the language used within the system under test with respect to old studies for prioritization of test cases.

3.6 Year 2005
A Controlled Experiment Assessing Test Case Prioritization Techniques via Mutation Faults

In [7] Hyunsook Do and Gregg Rothermel suggests that real fault may be represented by mutation faults. A controlled experiment has been performed to determine the ability of TCP techniques for the improvement in the rate of fault detection in relation to mutation faults as well as real faults. The result shows that TCP techniques works effectively while assessing mutation faults. The effectiveness of TCP techniques varies according to the characteristics of test cases and mutant faults. Also, comparison of relationship between mutation faults and hand seeded faults along with earlier data has been done.

3.7 Year 2006

3.7.1 On the Use of Mutation Faults in Empirical Assessments of Test Case Prioritization Techniques

In [8] Hyunsook Do and Gregg Rothermel carried out the empirical study which suggests that the real faults may be represented via mutation fault. Also it is suggested that hand seeded faults may cause problem for the authenticity of empirical result that focus on the rate of fault detection of test suites. The two controlled experiment has been performed to determine the ability of TCP techniques relative to the mutation faults.

3.7.2 Cost-Cognizant Test Case Prioritization

In [9] Alexey G.Malishevsky et al presents a new metric for determining the rate of fault detection of TCP, $APFD_c$. The metric defined include varying test case cost as well as varying fault cost. This paper also describes how the previous prioritization techniques work well with the new cost-cognizant factor. The study shows the comparison between

the cost-cognizant techniques as well as the cost-cognizant metric to that of previous non-cost-cognizant techniques and metric. This study gives consideration to answer the practical question of real world scenario.

3.8 Year 2008
Configuration-Aware Regression Testing: An Empirical Study of Sampling and Prioritization

In [10] Xiao Qu et al addresses the shortcomings of the configuration aware regression testing for the evolvement of software systems. The study makes use of combinatorial interaction testing techniques for modelling and generating the configuration samples that were used during the regression testing process. An empirical study was conducted over the non-trivial evolved software system that evaluates the influence of configuration over testing effectiveness. The result shows that there was significant impact of configuration on the rate of fault detection and also, prioritizing the configuration was effective.

3.9 Year 2010
The Effects of Time Constraints on Test Case Prioritization: A Series of Controlled Experiments

In [11] Hyunsoo Do et al suggested that imposing the time constraints over the process of regression testing by the software development process may significantly affect the behaviour of TCP techniques. The study shows that effective application of time constraint over the prioritization techniques may improves the rate of fault detection, software maintenance and testing process. A series of experiment has been conducted to determine the effects of time constraints over the costs as well as the benefits of the prioritization techniques. The results show the various implications for the system engineer who wish to retest the software system cost effectively.

3.10 Year 2011
A Model Based Prioritization Technique for CBSS Retesting Using UML State Chart Diagram

In [12] Sanjukta Mohanty et al Proposes a new efficient technique to prioritize the test cases for the CBSS using regression testing. In this technique, developer construct UML state chart diagram for all components and change of state by the different modules in the CBSS. The UML chart diagrams are then converted into Graph like representation known as component interaction graph (CIG). These graphs help in determining the interrelationship that exists in between the components. The new proposed algorithm takes the UML generated graph as an input along with the old test cases. The output of the algorithm is the efficient prioritization of test cases. The prioritization of test cases is done on the basis of two factors i.e. (1) total number of database access and (2) the number of state changes by the components when interacted with test cases. Also, the algorithm works efficiently for many java applications by enhancing the performance function and decreasing the cost.

3.11 Year 2012

3.11.1 Oracle-Centric Test Case Prioritization

In [13] M. Staats et. al. propose a technique for the TCP that explicitly considered the impact of test oracles to measure the effectiveness of testing. In the existing work, effect of test oracle has not been considered during retesting process. The new technique works by getting the flow information from the variable assignment and thus test oracle for individual test case. Later, prioritizing of cover variable was done with the help of shortest path available to test oracle. The results show that there is significant enhancement in the rate of fault detection in relation to both structural and random coverage based TCP techniques.

3.11.2 Modular Based Multiple Test Case Prioritization

In [14] N.Prakash and Rangaswamy proposes a new technique that prioritizes the test cases at modular level, to alleviate the problem of cost and time. The existing techniques for TCP consumes more time and cost, however not reliable and efficient. In this new proposed technique the program is first decomposed into different modules and then generate the test case for each module, in the first stage the test case corresponding to each individual module is prioritized and then in second stage the individual test suit are recombined together to further prioritizes the whole program. And also this technique is verified for fault coverage, moreover compared with overall test case periodization method.

3.12 Year 2013
Bridging the Gap between the Total and Additional Test Case Prioritization Strategies

In [15] L. Zhang et al proposes two different models that unify the two well-known TCP strategies i.e. the total TCP strategy and the additional TCP strategy. The two models i.e. basic model and extended model gives a spectrum of evasive strategy that ranges from total strategy to additional strategy, that depends on the specified parameter referred as x value. The four different heuristic have been taken to get the differentiated values of x for the different test methods. The empirical study has been performed over the 19 versions of given four Java program under test. The results show that differentiated value of x applied over the both basic and extended models using method coverage may performed better than the additional strategy with statement coverage.

3.13 Year 2014
A Unified Test Case Prioritization Approach

In [16] D. Hao et al presents an unified TCP approach that encircles both the total TCP strategy and additional TCP strategy. The unified TCP approach contains two models i.e. basic and extended model. Using these two models a unified TCP approach produces the spectrum of TCP techniques that ranges from entirely total to entirely additional TCP techniques defined by unique parameter z. To evaluate the approach, an empirical study has been performed over the 28 Java and 40 C objects. The result shows that numerous prioritization techniques derived from the two models with z value may perform better than entirely total TCP technique and also competitive with entirely additional TCP technique.

4. CONCLUSION

In this paper, a review on existing techniques of regression TCP based on code coverage, components and UML state chart diagrams is done. To evaluate the operative efficiency of

TCP techniques, two metrics named as APFD and APFDc are used. Also, paper presents how the TCP techniques evolve from year 1999 to present and how the prioritization techniques that were used only for C dialects evolved to be used in other language such as Java under JUnit framework and many more. Thus, it may be conclude that there are many techniques that may be used to prioritize the test cases in an efficient manner.

5. REFERENCES

[1] G. Rothermel, R. Untch, C. Chu, M. Harrold,"Test Case Prioritization: An Empirical Study,",Proc. IEEE International Conference on Software Maintenance, pages 179-188, 1999.

[2] S. Elbaum, A. Malishevsky, and G. Rothermel," Prioritizing Test Cases for Regression Testing", Proceedings of the ACM International Symposium on Software Testing and Analysis, pages 102-112, August 2000.

[3] S. Elbaum, D. Gable, and G. Rothermel, "Understanding and Measuring the Sources of Variation in the Prioritization of Regression Test Suites", Proceedings of the 7th International Software Metrics Symposium, pages 169-181, April 2001.

[4] S. Elbaum, A.Malishevsky, and G. Rothermel, "Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization", Proceedings of the 23rd International Conference on Software Engineering, IEEE Computer Society, pages 329-338, May 2001.

[5] S. Elbaum, A. Malishevsky, and G. Rothermel. "Test case prioritization: A family of empirical studies", IEEE Transactions on Software Engineering, 28(2), pages 159-182, February 2002.

[6] H. Do, G. Rothermel, and A. Kinneer, "Empirical Studies of Test Case Prioritization in a JUnit Testing Environment", Proceedings of the International Symposium on Software Reliability Engineering, pages 113-124, November 2004.

[7] Hyunsook Do and Gregg Rothermel, "A Controlled Experiment Assessing Test Case Prioritization Techniques via Mutation Faults", Proceedings of the IEEE International Conference on Software Maintenance, pages 411-420, 2005.

[8] Hyunsook Do and Gregg Rothermel, "On the Use of Mutation Faults in Empirical Assessments of Test Case Prioritization Techniques", IEEE Transactions on Software Engineering, V. 32, No. 9, pages 733- 752, 2006.

[9] Alexey G. Malishevsky, Joseph R. Ruthruff, Gregg Rothermel and Sebastian Elbaum, "Cost-cognizant Test Case Prioritization", Technical Report TRUNL-CSE-2006-0004, Department of Computer Science and Engineering, University of Nebraska – Lincoln, 2006.

[10] Xiao Qu, Myra B. Cohen and Gregg Rothermel, "Configuration-Aware Regression Testing: An Empirical Study of Sampling and Prioritization," In.Proc.of the 2008 international symposium on Software testing and analysis, pages 75-85, 2008.

[11] Hyunsook Do, Siavash Mirarab, Ladan Tahvildari and Gregg Rothermel, "The Effects of Time Constraints on Test Case Prioritization: A Series of Controlled Experiments", IEEE Transactions on Software Engineering archive, Vol. 36, No. 5, pages 593 - 617, Sep 2010.

[12] Sanjukta Mohanty, Arup Abhinna Acharya, Durga Prasad Mohapatra, "A Model based prioritization technique for Component based software retesting using UML state chart diagram", IEEE Third Int'l Conf. on Electronics Computer Technology, vol. 2, pages 36-368, 2011.

[13] M. Staats, P. Loyola, G. Rothermel, "Oracle-Centric Test Case Prioritization", Proceedings of the International Symposium on Software Reliability Engineering, pages 311-320, November 2012.

[14] N. Prakash and T. R. Rangaswamy, "Modular Based Multiple Test Case Prioritization", In Computational Intelligence and Computing Research, IEEE International Conference on, pages 1-7, 2012.

[15] L. Zhang, D. Hao, L. Zhang, G. Rothermel, H. Mei, "Bridging the Gap Between the Total and Additional Test-Case Prioritization Strategies", Proceedings of the International Conference on Software Engineering, pages 192-201, May 2013.

[16] D. Hao, L. Zhang, L. Zhang, G. Rothermel and H. Mei, "A Unified Test-Case Prioritization approach", ACM Transactions on Software Engineering and Methodology (TOSEM) 24, no. 2, 2014.