International Journal of Information Technology and Management
Vol. 12, Issue No. 1, February-2017, ISSN 2249-4510

Test Case Prioritization and Optimized Test Case Generation in Software Testing

Priyanka[1]* Shaista Khan[2] Khadija Sania Ahmad[3]

[1]M. Tech in CSE Scholar, Al-Falah University, Dhauj, Faridabad, Haryana

[2]M. Tech in CSE Scholar, Al-Falah University, Dhauj, Faridabad, Haryana

[3]M.Tech in CSE Scholar, Al-Falah University, Dhauj, Faridabad, Haryana

Abstract – The outline of a fitting test suite for software testing is a testing errand. It requires an appropriate tradeoff between viability, e.g., an adequate measure of experiments to fulfill the test objectives of a given scope rule, and productivity, e.g., a repetition diminished choice of experiments. Late test suite improvement approaches, along these lines, normally require an unequivocal count of existing experiments to choose from. The test suite outline for covering whole software product offerings is significantly more risky as the reliance between experiments, test objectives and item setups must be considered. Because of the exponential number of setups regarding the quantity of elements, an express list of all items for advancing a product offering test suite is impartibly. To handle this issue, we propose an incremental test suite enhancement approach for product offering testing that does not require an unequivocal portrayal of the arrangement of designs under test, yet rather utilizes a typical portrayal as far as highlight imperatives.

Keywords: Optimization, Testing and Software

----------X----------

1. INTRODUCTION

Automated software testing is an development in which software tools execute pre-scripted tests on a product application before it is discharged into creation. Automated testing apparatuses are capable of executing tests, revealing conclusions and contrasting outcomes and past trials. Tests completed with these devices can be run over and again, whenever of day. The technique or process being utilized to put into practice computerization is known as a test robotization system. A few structures have been acknowledged throughout the years by business merchants and testing associations. Robotizing tests with business off-the-rack (COTS) or open source software can be unpredictable, be that as it may, in light of the fact that they quite often require customization. In numerous associations, robotization is just put into honed when it has been unfaltering that the manual testing project is not meeting prospect and it is impractical to get more human analyzers.

It is a bungle to expect that test mechanization is just detained and play again of a manual test handle. Truth be told, computerization is fundamentally not quite the same as manual testing: there are altogether extraordinary issues and possibility. What's more, even the best computerization will never totally supplant manual testing, since mechanization is about certainty and clients are inalienably sporadic. Along these lines, utilize computerization to confirm what we expect, and utilize manual testing for what we don't.

A key believed is that we can't computerize an improvement that is not officially very much characterized. A completely manual process might not have the direction or documentation important to hold up very much outlined computerization documentation.

In any case, notwithstanding when the test procedure is consistently all around characterized, computerization is as yet a test. The reason of this report is to conquer any hindrance flanked by what ought to be tried and how it ought to be computerized. This starts by laying out certain fundamental rule that apply which must be comprehended before achievement is conceivable. These standards can be reiterated in one essential start: test product is programming!

Test computerization is truly two distinct things testing, which is one train, and mechanization, which is another. Mechanizing software testing is the same than robotize secretarial or whatever other business

work: for each situation, a PC is being told to play out an assignment before performed physically. Regardless of whether these orders are put away in something many refer to as a script or a program, they both have the greater part of the attributes of source code.

Application software must be expected to be viable over its helpful life, so should our computerized tests.

One reason practicality is so huge is that without it we can't develop tests. By and large, 25% of an application is revamped every year; if the tests connected with the adjusted bits can't be changed with a sensible measure of exertion, then they will be outdated. In this manner, as an option of bit by bit humanizing our test scope after some time by gathers increasingly test cases.

An amazing experiment ought to have the fabulousness to cover more elements of test goal. As such productivity of testing development depends on the nature of experiments not in the amount of experiments, which thus waits the testing time. We can get an appropriate sum number of experiments of better quality, by taking out repetitive experiments. So the issue of additional time use in testing stage can be lessened. Be that as it may, getting every one of those experiments in a surge time is a profound assignment. In this manner, programmed era of experiments decreases the exertion of an analyzer and designer thus cost and time. There are such a large number of move toward utilized for programmed experiment generation by utilizing transformative calculation calculations, yet they can't manage the excellent conduct of experiment. A decent experiment ought to test more than a certain something, in this manner falling the aggregate number of experiments required. Our proposed approach is more viable by covering what it is future to do as well as what it is not planned to do, by having the effect between these two conducts.

We utilize UML action charts and Collaboration outlines as plan stipulation and build up a move toward for experiment era. UML graphs clarify the diverse part of software frameworks relying upon the action to be performed. Plausibility is there that each of this part of the framework may make diverse sort of blunders. So it will be particularly valuable to utilize the both of the graph to handle each of these mistakes. Our approach utilizes hereditary calculation for era of enhanced experiments, because of its straightforwardness and viability.

2. REVIEW OF LITERATURE

- Most normal Key Performance Indicators (KPIs) in testing depend on the quantity of experiments as its key metric. Notwithstanding, the quantity of experiments does not express the ideal hazard scope that can be accomplished, since client studies have uncovered very repetitive experiments and holes in hazard scope.

- Browse an assortment of techniques to limit the quantity of experiments. Refine them promote with our licensed Linear Expansion system and robotized test suite set up s each experiment's commitment to hazard scope.

- Instinctive experiment definition, without the utilization of approach, is as yet the most widely recognized testing arrangement. Overviews uncover that this practice conveys reasonable experiment quality, with hazard scope of up to 40 – 50 %. Clients who endeavor to accomplish expanded hazard scope of 90%+ with natural experiment definition, frequently come up short since exponential development of number of experiments with high redundancies.

- Experiment support requires all adjustments in test target conduct to be reflected in experiments. Lithe improvement makes additionally challenges as a result of its dynamic nature and its steady condition of flux. Computerized test suite gives protest arranged ideas to limit these endeavors. Regardless of the possibility that clients prevail with regards to making fitting arrangements of experiments, frequently they neglect to keep up these advantages over a more extended timeframe, therefore decreasing the estimation of experiments and trading off the extensiveness of relapse tests.

- Programming application advancement is a predictable procedure requiring a great deal of changes and improvements every now and then. Nonetheless, not all engineers figure out how to make applications that impeccably meet their customers' assorted needs. As indicated by various reviews, over 70% of programming work identifies with application support and change.

- As the components or necessities change and the quantity of such changes develop, it gets to be distinctly troublesome for analyzers to perform relapse testing. Be that as it may, it is conceivable to mechanize relapse test suites. Mechanizing relapse test suites will empower the analyzers to accomplish better test scope all the time. Additionally, it empowers the assets to concentrate more on the more up to date and more convoluted application usefulness.

- In any case, analyzers regularly confront issues in the support of the computerized relapse test suites. There are a couple elements that influence the viability of the

relapse test suites. Among them are situations when test suite upgrades escape match up with whatever is left of the items or when tests are added to the suite with a fleeting viewpoint, accessibility of repetitive tests and that's only the tip of the iceberg.

Anyway, how to keep up the adequacy of the relapse test suite, or improve relapse test suite adequately? Here's the rundown of a few tips that may bail we out:

- Regression Test Selection (RTS) is a standout amongst the most well-known strategies for experiment suite enhancement. This strategy isolates the test suite into reusable experiments, re-testable experiments and out of date experiments. Aside from all these, it likewise makes new experiments that test the program for ranges not shrouded in current experiments.

- Another incredible approach to keep up the viability of the relapse test suite is to have a decent following system between the components a work in progress. This ought to be a reliable movement keeping in mind the end goal to keep up the test suites viably as it would help test director to confirm the component agenda and approve the test scope for an element that is being produced in the discharge.

- The viability of the relapse test suite can be effortlessly kept up by observing the progressions to the test suite. An unmistakably sketched out process will guarantee that exclusive tests that are valuable to the whole testing technique get added to the test suite, which guarantees the productivity and ease of use of the test bridle at an abnormal state.

- Considering intermittent cleanup of old tests is another awesome way to deal with keep up viability of an element rich relapse test suite. In this situation, all the current tests in the test suite should be broke down for their adequacy in a particular situation. Likewise, there will be situations where certain components won't be upheld in light of the diverse item course. In such cases, the significant relapse test suites ought to likewise be dialed down. It will guarantee power of the relapse test suite for a drawn out stretch of time.

- We can likewise quantify the viability of relapse test suites on a discharge to-discharge premise. It will permit us to know the main driver for diminishment in the adequacy of the test outfit assuming any, and empower us to make fitting move on the same.

- Collection of a few measurements and their investigation could likewise be helpful with regards to the adequacy of the relapse test suite. It will help we get great perceivability on the viability of the relapse test suite. We can consider diverse measurements, for example, rate of deformities found by the relapse tests suite, their significance, and so forth.

3. TEST CASE SELECTION

Experiment determination is a technique for choosing a subset of experiments from a test suite to lessen the time, cost and exertion in programming testing process. It is especially like test suite minimization method. The test suite minimization system depends on measurements like scope measured from a solitary form of the program under test. The distinction between these two strategies relies on the progressions made in SUT. The test cases are chosen by the changes made between the past and the present variant of the SUT.

In Model based procedure, the subset of the necessities gets displayed by utilizing formal documentations, for example, particulars of the subset of the prerequisites. (Leon *et. al.,* 2010). displayed an experimental examination of four unique strategies for separating huge test suites-test suite minimization, prioritization by extra scope, bunch sifting with one group inspecting, disappointment interest testing. Rothermal [35] also characterize a strategy for test suite minimization where the measure of the test can be decreased by disposing of excess experiments from the test suite. In this way minimization method is additionally called as test suite reduction .Leung and White present the first deliberate way to deal with relapse testing and test cases. Tallam et.al performed two sorts of diminishment on the lattice. Lattice is a characteristic portrayal that backings the distinguishing proof of the experiments. Test suite improvement issue is explained by executing the model based test suite advancement method, in this system the developmental based calculations are utilized for upgrading the test suite .Another procedure used to lessen the aggregate number of experiments are Extended Finite State Machine(EFSM) .It is fundamentally used to decrease the relapse test suites.

4. TEST CASE PRIORITIZATION AND EVALUATION

Experiment prioritization is a strategy for planning and positioning the experiments from various test

suites of programming. There are many ways to deal with timetable and rank the experiments. Every last experiment is doled out some need however in some cases there might be a few issues emerge when various experiments have a similar need or the weights. Once in a while issue happens in organizing these numerous test suites. There are two strategies to conquer these issues.

IT depicts the Multiple Test Suite prioritization (MTSP) strategy, which is utilized to organize the experiments from various test suites. The whole program is isolated into number of test suites and these test suites contains different experiments. The experiments are organized by weight and rank that are utilized for testing the program. Rothermal considered nine methodologies for organizing an arrangement of experiments and detailed outcomes. He likewise displayed distinctive methodologies for uncovering the flaws to enhance the product quality. Relapse testing is the way toward testing the product against those adjustments in the current programming. The four strategies for relapse testing are-reset technique, relapse test choice strategy, test suite lessening and experiment prioritization technique, displayed different experiment prioritization approaches that depend on a few criteria.

The main approach is Distribution-based approach, in which the experiment profiles are dispersed in view of the uniqueness metric. Utilizing this metric, the groups of these experiments are set up as indicated by their profiles. The experiments having comparable profiles get bunched into one repetitive gathering of experiments and different gatherings show the unordinary conditions that cause the experiment disappointments.

The second approach is Human based approach, which depends on Case-Based Reasoning (CBR) in which a Rankshoot calculation is taken that chooses test cases as indicated by their positions gave. Shin and Harman [38] gives some different methodologies where the experiments get organized by the likelihood of experiment determination methods.The test cases are chosen in light of a few elements like cost, length of experiments etc.The history-based approach is related with the bunches in view of some pervious curios that are gotten by lattice examination. Some experiments areprioritized as per the product necessities of the clients.

Theother approach is demonstrate based approach, where the applicable experiments are relegated into high and low need test cases based onthe outlined model.[Jiripong] depicts an investigation's plan, estimation measurements and results with a specific end goal to decide the most prescribed experiment prioritization strategy.

The experiments are assessed to survey and contrast the appropriate experiments with test the product. To assess the experiments, the accompanying experiment prioritization methods are utilized:

(1) Prepare try information

(2) Run the test suites prioritization strategy

(3) Evaluate comes about.

A few estimations measurements are additionally utilized as a part of this examination are:

(1) Percentage of high need save viability,

(2) Size of satisfactory experiments,

(3) Total prioritization time.

These techniques help in finding the base number of experiments for testing programming.

5. TEST CASE SELECTION AND OPTIMIZAION

Experiment choice is a technique for choosing a subset of experiments from a test suite to diminish the time, cost and exertion in software testing process. It is especially like test suite minimization system. The test suite minimization system depends on measurements like scope measured from a solitary adaptation of the program under test. The contrast between these two methods relies on the progressions made in SUT. The experiments are chosen by the progressions made between the past and the present variant of the SUT.

- **Model based Technique and Extended Finite State Machine (EFSM)**

In Model based procedure, the subset of the necessities gets displayed by utilizing formal documentations, for example, details of the subset of the prerequisites. Biswal, Baikuntha. Narayan, exhibited an experimental correlation of four unique strategies for sifting vast test suites-test suite minimization, prioritization by extra scope, bunch separating with one group inspecting, disappointment interest examining. P.D Ratna Raju, Suresh,Cheekaty, Harish Babu. Kalidasu , likewise characterize a method for test suite minimization where the span of the test can be decreased by disposing of repetitive experiments from the test suite. In this manner minimization technique is likewise called as test suite decrease. Leung and White present the primary efficient way to deal with relapse testing and experiments. Tillmann et.al performed two sorts of diminishment on the cross section. Cross section is a characteristic portrayal that backings the recognizable proof of the experiments. Test suite improvement issue is tackled by actualizing the model based test suite enhancement procedure. In this procedure the transformative based calculations are utilized for

upgrading the test suite .Another strategy used to decrease the aggregate number of experiments are Extended Finite State Machine(EFSM) .It is essentially used to diminish the relapse test suites.

6. TEST CASE PRIORITIZATION AND EVALUATION

Experiment prioritization is a technique for planning and positioning the experiments from various test suites of software. There are many ways to deal with timetable and rank the experiments. Every single experiment is appointed some need yet in some cases there might be a few issues emerge when different experiments have a similar need or the weights. In some cases issue happens in organizing these different test suites. There are two techniques to defeat these issues.

- **Multiple Test Suite prioritization (MTSP) method**

Legeard, depicts the Multiple Test Suite prioritization (MTSP) strategy, which is utilized to organize the experiments from various test suites. The whole program is partitioned into number of test suites and these test suites contains various experiments. The experiments are organized by weight and rank that are utilized for testing the program. Biswal considered nine methodologies for organizing an arrangement of experiments and announced outcomes. He additionally exhibited diverse methodologies for uncovering the shortcomings to enhance the product quality. Relapse testing is the way toward testing the product against those adjustments in the current programming. The four strategies for relapse testing are-reset technique, relapse test determination strategy, test suite lessening and experiment prioritization technique. Pakinam, introduced different experiment prioritization approaches that depend on a few criteria. The main approach is Distribution-based approach, in which the experiment profiles are disseminated in light of the divergence metric. Utilizing this metric, the groups of these experiments are set up as per their profiles. The experiments having comparative profiles get bunched into one excess gathering of experiments and different gatherings show the bizarre conditions that cause the experiment disappointments.

The second approach is Human construct approach which is situated in light of Case-Based Reasoning (CBR) in which a Rankshoot calculation is taken that chooses test cases as indicated by their positions gave. Pakinam, gives some different methodologies where the experiments get organized by the likelihood of experiment determination methods.The test cases are chosen in view of a few variables like cost, length of experiments etc.The history based approach is related with the bunches in light of some pervious relics that are gotten by grid investigation. Some experiments areprioritized as indicated by the product necessities of the clients.

- **Model based approach**

The other approach is demonstrate based approach, where the pertinent experiments are allocated into high and low need test cases in light of the outlined model, portrays a trial's plan, estimation measurements and results keeping in mind the end goal to decide the most suggested experiment prioritization technique.

The experiments are assessed to evaluate and contrast the appropriate experiments with test the product. To assess the experiments, the Following experiment prioritization systems are utilized :

(1) Prepare try information,

(2) Run the test suites prioritization strategy,

(3) Evaluate comes about.

A few estimations measurements are likewise utilized as a part of this investigation are:

(1) Percentage of high need hold viability,

(2) Size of adequate experiments,

(3) Total prioritization time.

These techniques help in finding the base number of experiments for testing programming. The table of experiment era system positioning is additionally appeared in that paper in light of these prioritization strategies with the assistance of Random approach, Hema's approach, Alexey's technique, MTSSP and MTSPM techniques for experiment era.

7. CONCLUSION

In this paper, we have primarily focused on test case generation of object-oriented software automatically. We have likewise investigated the strategy for use of developmental calculation like hereditary calculation to the programmed approach of testing. In this review we have proposed a way to deal with produce test cases for protest arranged projects by utilizing UML movement outlines. We have utilized a heuristic run to get the decreased experiments, which fulfill way scope as the test sufficiency criteria. In this part we have considered just the basic way for programmed experiment era. Our approach accomplishes the

greatest branch scope and way scope, which is an additional preferred standpoint.

Our approach is not appropriate to deal with the extensive and complex framework. This approach is particularly appropriate for basic frameworks where no more fork-joins, as settled fork joins and so forth are included, which is our next goal. However our proposed framework is not adequate to deal with various sort of blunders, for example, work process mistakes, state based blunders and so forth. To conquer this bottleneck, a joined approach is basic and henceforth we have utilized the numerous UML charts, for example, Activity, Class and Sequence graph. For our approach we have considered both Activity chart and Collaboration outline and we call them "air conditioning graph".

REFERENCES

Abdurazik and J. Offutt, “Using uml collaboration diagrams for static checking and test generation,” in Proceedings of the third International Conference on the UML.

Arcuri and X. Yao (2008). “Search based software testing of object-oriented containers”, Information Sciences, vol. 178, no. 15, August, pp. 3075-3095.

B. N. Biswal, S. S. Barpanda and D. P. Mohapatra (2010). International Journal of Computer Applications, vol. 1, Issue 14.

Binder R.V. (2000). Testing Object Oriented Systems:Models,Patterns and Tools. Addison Wesley: USA , 2000.

Biswal, Baikuntha. Narayan et al, (2010). Test Case Generation and Optimization of Object Oriented Software using UML Behavioral Models, NIT, Rourkela, Orissa, India, July, 2010.

Biswal, Baikuntha. Narayan et al, (2010). Test Case Generation and Optimization of Object Oriented Software using UML Behavioral Models, NIT,Rourkela, Orissa, India, July, 2010.

C. Mingsong, Q. Xiaokang, and L. Xuandong, (2006). “Automatic test case generation for UML activity diagrams,” in Proceedings of the 2006 international workshop on Automation of software test, Shanghai, China, pp. 2 – 8.

G. Booch, J. Rumbaugh, and I. Jacobson (2001). The Unified Modeling Language User Guide. Addison-Wesley.

G. Booch, J. Rumbaugh, and I. Jacobson, (2001). The Unified Modeling Language User Guide. Addison-Wesley.

G. Dunwei, Z. Wanqiu and Z. Yan (2011). Chinese Journal of Electronics, vol. 19, no. 2.

J. A. Jones, M. J. Harrold, (2003). Test Suite Reduction and Prioritization for Modified Coverage, IEEE Transactions on Software Engineering, 29(3), Pages 195-209, March 2003.

Legeard B. (2010). Model-Based Testing: Next Generation Functional Software Testing. Proceedings of practical Software Testing: Tool Automation and Human Factors Seminar. Published by :SchlossDagstuhl_ Leibniz_ZentrumfuerInformatik : Dagstuhl, Germany, 2010.

M. Prasanna, S. N. Sivanandam, R. Venkatesan, R. Sundarrajan (2005). “A Survey on Automatic Test Case generation”, Academic Open Internet Journal, vol. 15.

Object management group (2003). available at http://www.omg.org/uml.

P. McMinn (2004). “Search-based software test data generation: A survey”, Software Testing, Verification & Reliability, vol. 14, no. 2, June, pp. 105–156.
Sharma, A. Jadhav, P. R. Srivastava and R. Goyal, “Test cost optimization using tabu search”, J. Soft. Eng. Appl., vol. 3, no. 5, (2010), pp. 477–486.

P.D Ratna Raju, Suresh Cheekaty, Harish Babu. (2011). Kalidasu, object Oriented Software Testing, International Journal of Computer Science and Information Technologies, Vol.2(5), ISSN: 0975-9646, pg2189-2192.

Pakinam N. Boghdady, NagwaL. Badr, Mohamed Hashem, Tolba F. Mohamed A. (2011). Proposed Test Case Generation Technique Based on Activity Diagrams, International Journal of ngineering and Technology, IJETIJENS,ISSN: pp. 114703-5858, Vol:11, No:03, June 2011.

Q. Li and J. Li, (2009). Proceedings of the International Symposium on Intelligent Information Systems and Applications.

Q. Li and J. Li., (2009). Proceedings of the International Symposium on Intelligent Information Systems and Applications.

R. E. Prather and J. P. M. Jr. (1987). “The path prefix software testing strategy,” IEEE Transactions on Software Engineering, vol. 13, no. 7, pp. 761–766.

S. J. Cunning and J. W. Rozenblit (2005). “Test scenario generation from a structured

requirements specification", journal of Intelligent and Robotic Systems, vol. 41, no. 2-3, pp. 87-112.

S. K. Swain, D. P. Mohapatra and R. Mall (2010). "Test case generation based on state and activity models", Journal of Object Technology, vol. 9, no. 5, pp. 1 – 27.

Tillmann N and De Halleux J.Pex__ White Bor. (2008). Test Generation for .NET. In proceedings of the second Conference on Tests and Proofs (TAP) : Prato, Italy, 2008.

V. Rajappa, A. Biradar, S. Panda (2008). "Efficient software test case generation using genetic algorithm based graph theory", Proceedings of the First International Conference on Emerging Trends in Engineering and Technology, pp. 298-303.

W. H. Deason, D. B. Brown, K. H. Chang, and J. H. C. (1991). II, "A rule-based software test data generator," IEEE Transactions on Knowledge and Data Engineering, vol. 3, no. 1,pp. 108 – 117.

Yoo Shin (2001). Minimization, Selection and Priortization: A Survey, King's College London, Centre for Research on Evolution, Search and Testing, Strand, London, WC2R 2LS, UK.

York, U.K. (2000). Lecture Notes in Computer Science, Springer-Verlag GmbH, pp. 383 – 395.

Corresponding Author

Priyanka*

M. Tech in CSE Scholar, Al-Falah University, Dhauj, Faridabad, Haryana

E-Mail – priyankajmi17@gmail.com