Re-usability of Constraints for Test Case Generation in Different Applications using Genetic Algorithm

Sarbjot Kaur **Makul Mahajan**
Science & Engineering, Lovely Professional University, India

ABSTRACT
Software engineering is the study and an application of engineering to the design, development, and maintenance of software. Web testing is the name given to software testing that focuses on web applications. Regression testing means re-testing an application after its code has been modified to verify that it still functions correctly. In this test cases that have been develop using reusability constraints that must be act as priority based test cases for regression testing. To execute these cases on the basis of priority of area coverage adaptive TCP algorithm is used that provide as sequence of execution. This sequence has to be optimized by Genetic algorithm which uses population size, crossover and mutation probability for generation of new Childs.

Keywords:
Software Engineering, Web Testing, Regression Testing.

1. Introduction

1.1 Software Engineering

Programming designing is the study and a use of building to the outline, advancement, and support of programming. The Bureau of Labor Statistics' definition is "Exploration, plan, create, and test working frameworks level programming, compilers, and system dissemination programming for therapeutic, modern, military, interchanges, aviation, business, experimental, and general figuring applications." Software planning is the study and a utilization of building to the framework, change, and backing of programming. Regular formal implications of programming outlining are the utilization of an orderly, prepared, quantifiable route to the change, operation, and backing of programming a planning show that is concerned with all parts of programming generation additionally the establishment and usage of sound outlining models to financially procure programming that is reliable and satisfies desires gainfully on veritable machines. In view of inventive improvement and forcefulness in business, programming keeps developing.

1.2 Web Testing

Web testing is the name given to programming testing that spotlights on web applications. Complete testing of an online framework before going live can help location issues before the framework is uncovered to people in general. Issues, for example, the security of the web application, the essential usefulness of the website, its openness to impeded clients and completely capable clients, and additionally status for expected activity and number of clients and the capacity to survive a monstrous spike in client movement, both of which are identified with testing. Web testing is the name given to programming testing that spotlights on web applications. Complete testing of an online framework before going live can help location issues before the framework is uncovered to general society. Making a site does not end with putting all the media and programming together.

1.3 Regression Testing

Regression testing means re-testing an application after its code has been altered to check that despite everything it works accurately. Regression testing comprises of re-running existing experiments and watching that code changes did not break any already living up to expectations capacities, incidentally present mistakes or cause prior settled issues to return. Regression testing is characterized as the methodology of retesting and approving the altered parts of the product. Its principle objective is to guarantee that no new slips have been brought into already tried code or programming framework after changes have been made in the product. One of the principle purposes behind relapse testing is to figure out if a change in one piece of the product influences different parts of the product. Regular systems for relapse testing incorporate rerunning already finished tests and checking whether program conduct has changed and whether beforehand altered flaws have re-rose.
Regression testing can be performed to test a framework proficiently by methodically selecting the proper least arrangement of tests expected to sufficiently cover a specific change. Stand out from non-relapse testing (for the

Manuscript received May 5, 2016
Manuscript revised May 20, 2016

most part acceptance test for another issue), which intends to confirm whether, in the wake of presenting or overhauling a given programming application, the change has had the proposed impact.

Regression testing is utilized not just for testing the rightness of a system, yet regularly additionally to follow the nature of its yield. The reason behind relapse testing is exceptionally straightforward and straight. It expands the profitability and effectiveness of programming items and quality confirmation applications.

1.4 Types of Regression Testing:

1.4.1 Final Regression Tests: - A "final regression testing" is performed to validate the build that hasn't changed for a period of time. This build is deployed or shipped to customers.

1.4.2 Regression Tests: - A normal regression testing is performed to verify if the build has NOT broken any other parts of the application by the recent code changes for defect fixing or for enhancement.

1.5 Test Cases

An examination, in programming building, is an arranged of conditions under which an analyzer will make sense of if an application, programming system or one of its characteristics is working as it was at first settled for it to do. The framework for making sense of if an item extends or structure has easily gotten through or failed. In a couple of settings, a prophet could be a need or usage case, while in others it will be a heuristic. It may take different examinations to find that a thing structure or structure is considered sufficiently inspected to be discharged. Trials are a great part of the time proposed as test scripts, especially when made - when they are generally aggregated into test suites

1.6 Types of Test Cases

1.6.1 Formal test cases: With a specific end goal to completely test that all the necessities of an application are met, there must be no less than two experiments for every necessity: one positive test and one negative test. In the event that a prerequisite has sub-prerequisites, every sub-necessity must have no less than two experiments. Staying informed regarding the connection between the necessity and the test is regularly done utilizing a traceability framework.

1.6.2 Informal test cases: Experiments are not composed at everything except rather the exercises and results are accounted for after the tests have been run. In situation testing, speculative stories are utilized to help the analyzer thoroughly consider a complex issue or framework. These situations are generally not recorded in any subtle element. They can be as basic as a graph for a testing situation or they could be a portrayal written in exposition. The perfect situation test is a story that is rousing, trustworthy, complex, and simple to assess. They are typically not the same as experiments in that experiments are single steps while situations cover various ventures of the key.

2. Literature Survey

Reetika Nagar et al [1] "Introduction of Software Maintenance Testing" Support of programming is an extremely critical and imperative assignment however it is an exceptionally costly process. In this way, Software Maintenance Testing is fundamental amid programming testing stage. Deformities and mistakes found amid testing procedure must experience a re-test process so that blemishes, that is, imperfections and slips can be dispensed with effectively. Thusly, experiments are totally expected to advance and alter as per the evolving prerequisites. In this paper, a brief presentation of a few support testing's, for example, affirmation testing and relapse testing; and presentation of upkeep testing strategies has been given. These methods are to be utilized as a part of testing amid programming advancement to make testing methodology successful.

Thillaikarasi Muthusamy et al [2] "A New Effective Test Case Prioritization for Regression Testing taking into account Prioritization Algorithm" Regression Testing is the procedure of executing the arrangement of experiments which have gone on the past form or arrival of the application under test so as to approve that the first highlights and capacities are as yet filling in as they were beforehand. It is impracticable and in-sufficient assets to re-execute each experiment for a system if changes happen. This issue of relapse testing can be understood by organizing experiments. A relapse experiment prioritization strategy includes re-requesting the execution of test suite to build the rate of shortcoming discovery in prior phases of testing procedure. In this paper, experiment prioritization calculation is proposed to distinguish the extreme blames and enhance the rate of deficiency discovery. This proposed experiment prioritization calculation organizes the experiments taking into account four gatherings of pragmatic weight variable, for example, client allocated need, engineer watched code execution intricacy, changes in prerequisites, flaw effect, fulfillment and traceability.

Bharat Choudhary et al [3] "An approach of Regression testing for Service Oriented Architecture" Administration Oriented Architecture (SOA) is an application structural engineering in which administrations are portrayed utilizing a depiction dialect. It has changed the way business ventures with innovation with a quick request of re-arrangement time short. Other than this the testing of SOA is additionally a testing element for execution of big business structural engineering. Segment testing and mix are on key testing technique. Then again the relapse testing is distinguishing accidental mistakes that may have been presented as a consequence of changing a project module. In this paper we characterize a methodology of relapse testing for Service Oriented Architecture.

Marback, A. Et al [4] "An Effective Regression Testing Approach for PHP Web Applications". In this paper, Author propose another relapse testing approach that is connected to oftentimes fixed web applications, considering security issues, and specifically, concentrating on PHP programs. Our methodology recognizes the influenced regions by code changes utilizing effect investigation and produces new experiments for the affected ranges by changes utilizing system cuts considering both numeric and string information values. To encourage our methodology, we actualized a PHP Analysis and Regression Testing Engine (PARTE) and performed a controlled trial utilizing open source web applications. The outcomes demonstrate that our methodology is successful in decreasing the expense of relapse testing for often fixed web applications.

Chaturvedi, A et al [5] "A tool supported approach to perform efficient regression testing of web services" in this paper, Author exhibit an instrument bolstered way to deal with perform proficient relapse testing of web administrations. Utilitarian and non-useful web administration testing is finished with the assistance of WSDL parsing and relapse testing is performed by recognizing the progressions made from there on. We distinguish, order, and catch the web administration relapse testing needs into three distinct classes, to be specific, changes in WSDL, changes in code, and particular re-testing of web administration operations. To catch over three progressions we proposed three middle of the road manifestations of WSDL, to be specific, Difference WSDL (DWSDL), Unit WSDL (UWSDL), and Reduced WSDL (RWSDL), separately. These middle manifestations of WSDLs are then consolidated to shape Combined WSDL (CWSDL) which is further utilized for relapse testing of the web administration.

Hossain, M et al [6] "Regression Testing for Web Applications Using Reusable Constraint Values" Organizations that give web applications need to perform successive relapse testing in light of the fact that organizations regularly experience different security assaults and incessant highlight upgrade requests from clients. Ordinarily, such applications oblige relapse testing courses of action that oblige negligible test exertion on the grounds that they have as of now been conveyed and utilized as a part of the field.

In our past work, we introduced a proficient relapse testing approach that permits us to concentrate on the territories of code that have been changed and to relapse test them to address this issue. While our trial results demonstrated that this methodology can be proficient in sparing the expense of relapse testing by lessening the quantity of test ways essential for the altered system, Author additionally discovered that determining info imperatives obliges a considerable measure of exertion. In this paper, to oblige further reserve funds with relapse testing, we exhibit a strategy that recognizes reusable imperative qualities for relapse experiments.

3. Problem Formulation

Web testing is used widely in the various fields of computers and business. Due to changes in various fields of web applications the testing has to be performed again and again. A company that provides web applications undergoes different security attacks and feature changes in their products according to demands of users. Due to changes various patches undergoes under different regression testing approaches. Regression testing can be done on the basis of different types of changes made in entire project. Test suite has to be developed for the regression testing purpose. The test case used for the regression testing has been reused to save time and effort required to develop new test cases. The constraints value used in test cases get reused and these cases have been used for web testing. The main problem occurred in reusability of test cases is that it does not consider optimization of test suites developed automatically. The test cases generated has been optimized using expert system. In this test cases that have been develop using reusability constraints that must be act as priority based test cases for regression testing. Test suites must contain all subset to test previous versions as well as new changes done in these cases.

4. Methodology

In the present work, the constraints from both the version are checked if they can be reused. Checksim algorithm is used to find out the reusable constraints. Test cases are generated by using reusable constraints. Test Case execution sequence is generated on how to run the test cases with full efficiency. Further Genetic Algorithm is used to find out the best test case execution sequence. Lastly Parameters are computed such as reusable constraints, time and cost.

5. Result

Table 1: Result Table

Version Pair	0 & 1	0 & 1	0 & 1
Total Path	8	6	10
Total Inputs	4	3	3
Reusable Inputs	3	3	2
Regression Paths	11(1)	2(19,21)	3(13,14,17)
Reusability	75%	100%	66%

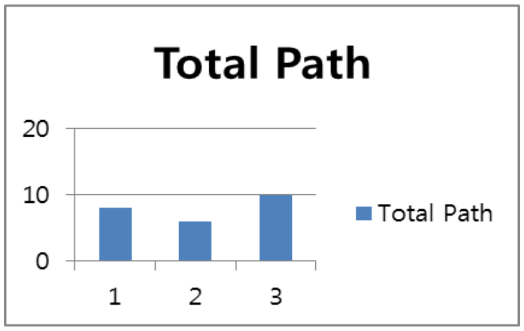

Figure 1: Total Number of Paths

This Graph represents total number of executable path in the code. In the first pair there are 8 test paths whereas the second pair consists of 6 paths and the third pair consist the highest number of test paths that is 10.

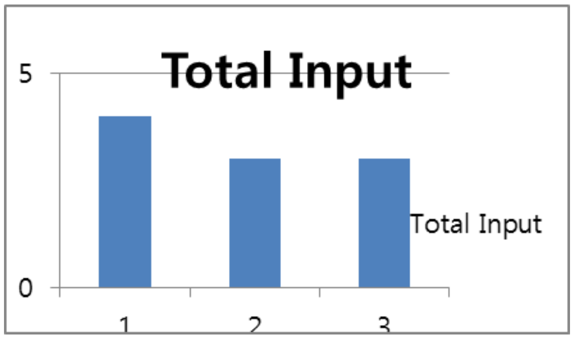

Figure 2: Total Number of Inputs

This graph represents total number of inputs. In the first pair there are 4 input values that are needed to be provided whereas the second and third pair needs 3 inputs.

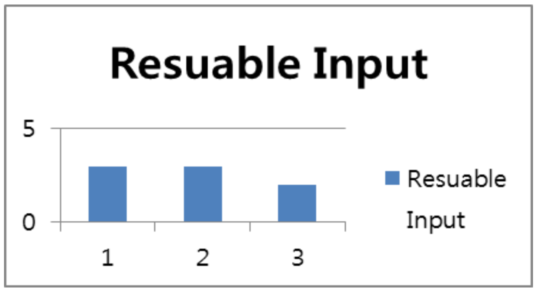

Figure 3: Reusable Inputs

This graph represents the main result that is in the first and second pair 3 inputs can be reused whereas 2 inputs can be reused in the third pair.

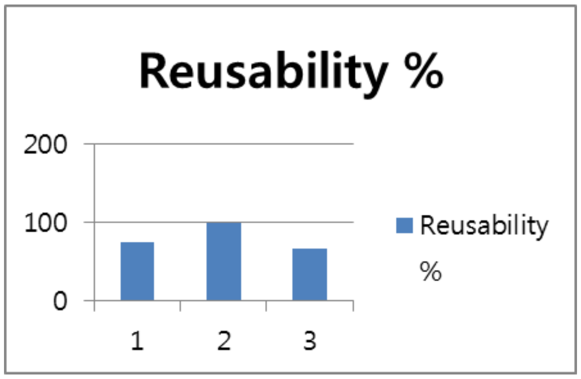

Figure 4: Reusability

This graph represents reusability percentage of the input values. Reusability percentage can be calculated by total number of inputs divided by reusable inputs. The first pair provides 75% of reusability and third pair 66% whereas second pair gives 100% reusability of its input values.

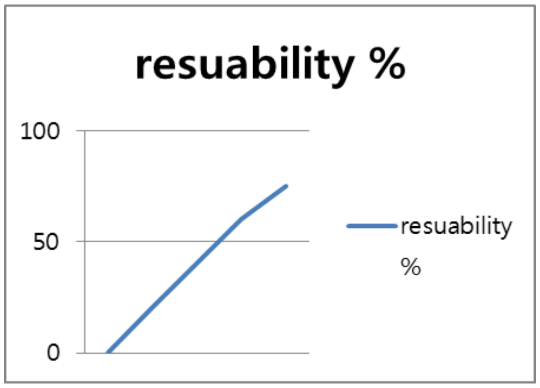

Figure 5: Reusability if 1st File

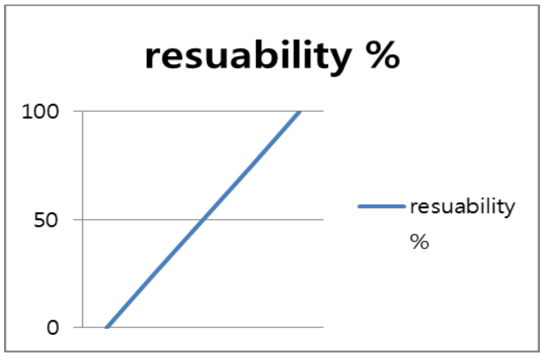

Figure 6: Reusability of 2nd File

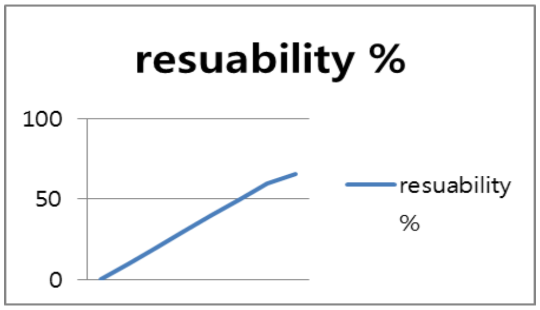

Figure 7: Reusability of 3rd File

6. Conclusion

Software engineering is the branch of computers to develop platform for various applications to be execute. In purposed work these cases has been designed on the basis of functionality and reusability of the constraint has been done so that that takes minimum time for generation of test case. Check-Sim algorithm is used to match the constraint values. On the basis of these values the constraints have been reused. After reusability of these constraints, theses test cases have to be executed for testing for checking the functionality. To execute these cases on the basis of priority of area coverage adaptive TCP algorithm is used that provide as sequence of execution. This sequence has to be optimized by Genetic algorithm which uses population size, crossover and mutation probability for generation of new Childs. For performance analysis of the purposed system parameters has been evaluated, these parameter are time, percentage of reusability are computed. These parameters have been compared with previous result that concludes that our approach is better than previous one.

7. FUTURE SCOPE

In the future work test cases reusability can be used in various real time applications. This reusability can be done by different algorithm and test case sequence can be optimized by other artificial intelligence approach.

REFRENCES

[1] Reetika Nagar “Introduction of Software Maintenance Testing”, ISSN 2277 128X, IEEE, 2014.
[2] Thillaikarasi Muthusamy “A New Effective Test Case Prioritization for Regression Testing based on Prioritization Algorithm”, ISSN 2249-0868, IEEE, 2014.
[3] Bharat Choudhary “An approach of Regression testing for Service Oriented Architecture”, ISSN 87685, IEEE, 2014.
[4] Marback, A. “An Effective Regression Testing Approach for PHP Web Applications”, ISSN 978-1-4577-1906-6, pp 221 – 230, IEEE, 2014.
[5] Chaturvedi, A. “A tool supported approach to perform efficient regression testing of web services”, ISSN 2326-6910, IEEE, pp 50 – 55, IEEE, 2013.
[6] Hossain, M “Regression Testing for Web Applications Using Reusable Constraint Values” ISSN 14351440, pp 312 – 321, IEEE, 2014.
[7] Wenhong Liu “Research and Application of Regression Test Case Design Methods Based on the Analysis of the Relationship” ISSN 13874437, pp 233 – 236, IEEE, 2013
[8] McMaster, S “Developing a Feedback-Driven Automated Testing Tool for Web Applications” ISSN 978-1-4673-2857-9, pp 210 – 213, IEEE, 2012.
[9] Leotta, M. “Capture-replay vs. programmable web testing: An empirical assessment during test case evolution”, ISSN 13916982, ISSN 13916982, pp. 272 – 281, IEEE, 2013.
[10] Lei Xu “A framework for Web applications tests”, ISSN 0-7695-2140-1, pp. 300 – 305, IEEE, 2014.
[11] N. Prakash1 “Modular Based Multiple Test Case Prioritization N. Prakash1,” International Conf. on Computational Intelligence & Computing Research (ICCIC), 2012, pp 1 – 7.
[12] Dan Hao “Adaptive Test-Case Prioritization Guided by Output Inspection” International conf. on Computer Software and Applications Conference (COMPSAC), 2013, pp 169 – 179.
[13] Xiaolin Wang “Dynamic test case prioritization based on multi-objective” IEEE Conf. on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 2014, pp 1 – 6.
[14] Chu-Ti Lin “History-Based Test Case Prioritization with Software Version Awareness” IEEE Conf. on Engineering of Complex Computer Systems (ICECCS) 2013, pp 171 – 172.
[15] Bo Jiang “Bypassing Code Coverage Approximation Limitations via Effective Input-Based Randomized Test Case Prioritization” IEEE Conf. on Computer Software and Applications Conference (COMPSAC), 2013, pp 190 - 199.