A Review on Various Techniques for Regression Testing and Test Case Prioritization

Jaspreet Singh Rajal
Assistant Professor
Department Of Computer Science & Engineering
Chandigarh University, Gharuan, Punjab, India

Shivani Sharma
Research Scholar, Master of Engineering
Department of Computer Science & Engineering
Chandigarh University, Gharuan, Punjab, India

ABSTRACT
In the field of software engineering, different applications have been developed. An application requires changes due to changes in the customer requirements. Regression testing has to be performed for the validation of data modification. Various test cases have to be developed to perform the regression testing. In this paper, various test case prioritization techniques have been discussed for the generation of priority of test suites and regression testing approaches provide information about which strategies have to be followed or not.

Keywords
Test Case, Prioritization, FEP, Test Suite, Regression Testing.

1. INTRODUCTION

1.1 Software Engineering
Software is instructions (computer programs) that when executed provide desired functionality and performance[1]. A good software must provide the required functionality, performance to the user and should be maintainable, reliable and usable. Software may be developed for a particular customer or may be developed for a general market. Computer science focuses on theory and fundamentals but software engineering is concerned with the practicalities of developing and delivering useful software. Software Engineering is the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software;[1]. While all softwares have to be professionally managed and developed, different techniques are appropriate for different types of system. For example, games should always be developed using a series of prototypes whereas safety critical control systems require a complete and analyzable specification to be developed. Software testing is a comprehensive set of activities conducted with the intent of finding errors in the software. The changed programming must be tried completely with the purpose so that the changed bit of code does not influence different parts of the code. Regression testing involves running test sets that have successfully executed after changes have been made to the system. The regression test checks that these changes have not introduced new bugs into the system and that the new code interacts as expected with the existing code.[2]. Common methods of regression testing include rerunning previously completed tests and checking whether program behavior has changed and whether previously fixed faults have re-emerged. Regression testing can be performed to test a system efficiently by systematically selecting the appropriate minimum set of tests needed to adequately cover a particular change.
Regression testing is important because changes and error corrections tend to be much more error prone than the original program code (in much the same way that most typographical errors in newspapers are the result of last-minute editorial changes, rather than changes in the original copy).It is a critical and exceptionally testing assignment for the product analyzers to test the whole programming inside the restricted time and assets. Every business in the United States now depends on software for the development, production, distribution and after sales support of products and services[3].A study conducted by NIST in 2002 reports that software bugs cost U.S. economy $59.5(Approx) billion annually[3].More than a third of this cost could be avoided if better software testing was performed [3].In case of complex software development, running the entire test cases require many weeks. Test engineers may want to prioritize and schedule those test cases in the order that those test cases with higher priority are executed first.

Test case prioritization methods and process are required, because[4]:

(a) the regression testing phase consumes a lot of time and cost to run

(b) there is not enough time or resources to run the entire test suite

(c) there is a need to decide which test cases to run first.

When the time required to execute all test cases in a test suite is short, test case prioritization may not be cost effective - it may be most expedient simply to schedule test cases in any order [4].When the time required to run all test cases in the test suite is sufficiently long, the benefits offered by test case prioritization methods become more significant.

Fig 1: Software Engineering

1.2 Regression Testing
Regression testing is frequently applied but it is expensive maintenance process that aims to (re)verify modified software. It is full or partial selection of already executed test cases which are re-executed to ensure existing functionalities

work fine. Regression testing does not belong to either unit test, integration test or system testing. Instead, it is a separate dimension to these three forms of testing[5]. If regression testing exposes problems, then test engineers check whether there are problems in the previous increment that the new increment has exposed or whether these are due to the added increment of functionality.

For example, there are three modules in the project named Admin Module, Personal Information and Employment Module and suppose bug occurs in the Admin Module like on Admin Module existing user is not able to login with valid login credentials so this is the bug. Now Testing team sends the above - mentioned bug to the development team to fix it and when development team fixes the bug and hand over to testing team. Testing team checks that fixed bug does not affect the remaining functionality of the other modules (Admin, PI, Employment) and also the functionality of the same module (Admin) so this is known as the process of regression testing done by software testers. Thus, regression testing assures programs not adversely affected by unintended modifications by running them over existing regression test suites. The information collected in the previous executions of the test cases (e.g., the fault history, the change history, and execution profiles) on the previous versions can be used to optimize the current round of regression testing. Basic techniques for regression testing incorporate rerunning already finished tests and checking whether program conduct has changed and whether beforehand altered shortcomings have re-developed. Regression testing can be performed to test a framework productively by deliberately selecting the fitting least set of tests expected to sufficiently cover a specific change.

1.2.1 Need Of Regression

Regression Testing is required when there is a

- Change in requirements and code is modified according to the requirement
- New feature is added to the software
- Defect fixing
- Performance issue fixing

1.2.2 Types Of Regression

Regression Testing can be classified as [6].

- *Local* - Changes introduce new bugs.
- *Unmasked* - Changes unmask previously existing bugs.
- *Remote* - Changing one part breaks another part of the program. For example, Module A writes to a database. Module B reads from the database. If changes to what Module A writes to the database break Module B, it is remote regression.
- *Corrective*

  It is applied when specifications are unmodified and test cases can be reused. It is triggered by corrections made to the previous version.
- *Progressive*

  It is applied when specifications are modified and new test cases must be designed. It is triggered by new features added to the previous version.

2. LITERATURE REVIEW

N.Prakash[7] "Modular Based Multiple Test Case Prioritization " concluded that cost and time effective reliable test case prioritization technique is the need for present software industries. Test case prioritization for the entire program consumes more time and the selection of test case for entire software is also affecting the test performance. In order to resolve it, author proposed modular based test case prioritization for regression testing. In this method the program is divided into multiple modules. The test cases corresponding to each module is prioritized first. In the second stage, the individual modular based prioritized test suites are combined together and further prioritized for the whole program. This method is verified for fault coverage and compared with overall program test case prioritization method. The proposed method is assessed using three standard applications namely University Students Monitoring System, Hospital Management System, and Industrial Process Operation System. The empirical studies conducted by the author showed that the proposed algorithm performed significantly well.

S.Raju,G.V.Uma[8]"Factors Oriented TestCase Prioritization Technique in Regression Testing using Genetic Algorithm", In this paper the regression testing based test suite prioritization technique is illustrated. A new prioritization technique is proposed for the requirement based system level test cases to improve the rate of fault detection of severe faults.

Chen L., Wang Z. et. al[9] "Test Case Prioritization for Web Service Regression Testing" proposed a dependence analysis based test case prioritization technique for Web Service regression testing. First, they analyzed the dependence relationship using control and data flow information in an orchestration language: WS-BPEL. Then they construct a weighted graph. After that, they prioritize test cases according to covering more modification-affected elements with the highest weight. Finally authors conduct a case study to illustrate the applicability of method.

Gaurav Duggal, Bharti Suri[10] "Understanding Regression Testing Techniques" described regression testing is done in the maintenance phase of the software development life cycle to retest the software for the modifications it has undergone. Approximately 50% of the software cost is involved in the maintenance phase. Thus, researchers are working hard to come up with best results by developing new regression testing techniques so that the expenditure made in this phase can be reduced to some extent. This paper discussed techniques of regression testing ,test case prioritization, importance of regression testing and its scope.

Shweta A. Joshi et al[11] "Literature Review of Model Based Test case Prioritization" described algorithms for prioritizing test cases for testing the component-based software systems. The studies show that testing of component-based software systems is expensive. The importance is given to component interactions because maximum defect occur when components are going to interact with each other. This approach is applicable to test the component composition in case of component based software maintenance.

S.Roongruangsuwan et.al[12] "Test-Case Prioritization techniques" researchers propose many methods to prioritize and reduce the effort, time and cost in the software testing phase, such as test case prioritization methods, regression selection techniques and test case reduction approaches. This

paper concentrates on test case prioritization techniques researched between 1998 and 2008.

This paper reveals that there are many research challenges and gaps in the test case prioritization area.

Animesh Caturvedi et al [13] "A Tool Supported Approach to Perform Efficient Regression Testing of Web Services" authors presented a tool for perform regression testing of web services. In this paper, functional and non-functional testing is performed by using WSDL and Regression testing is performed by identifying the changes made. Authors identify, categorize and capture the web service regression testing needs into three different categories, namely, changes in WSDL, changes in code and selective re-testing of web service operations. This paper proposed tool that can used to find out the exactly which operations are changed and how to use only those changed operations to minimize regression testing.

Lijun Mei,Zhenyu Zhang et al[14] "Test Case Prioritization for Regression Testing of Service-Oriented Business Applications", researchers have examined the important impact of heterogeneous artifacts on test case prioritization in the regression testing of service-oriented business applications and demonstrated the shortcomings of traditional test case prioritization techniques in this aspect. The proposed test case prioritization techniques take into account the coverage data of test cases at three levels i.e. workflow, XPath, and WSDL.

Sebastian Elbaum, Alexey G. Malishevsky[15] "Prioritizing Test Cases for Regression Testing", empirically examined the abilities of several test case prioritization techniques to improve the rate of fault detection of test suites. It focus on version-specific test case prioritization, in which test cases are prioritized and rate of fault detection is measured, relative to specific modified versions of a program.

Chu-Ti Lin et al [16] "History-Based Test Case Prioritization with Software Version Awareness" described that numerous existing prioritization techniques are code-based, in which the testing of every product form is considered as an autonomous procedure. The test after effects of the former programming renditions may be valuable for booking the experiments of the later programming forms. The paper described history-based ways to address this issue. History based prioritization is assigned to test case based upon the software versions and experimentation results of similar type of test cases.

Md. Hossain et al[17] "Regression Testing for Web Applications Using Reusable Constraint Values" described that companies often encounter various security attacks and frequent feature update demands from users so they need to perform frequent regression testing. Web applications require regression testing processes that require minimal test effort because they have already been deployed and used in the field. In this paper, researchers presented a technique that identifies reusable constraint values for regression test cases by analyzing definitions and uses of the variables for two consecutive versions. A large number of constraint input values may be reused from the previous version's test cases. The constraints and actual values for variables may be reusable across several versions as long as the definition and use relationships of the variables hold across versions. This will create greater savings as the applications evolve over time. It may also reduce testing effort as well as improve testing effectiveness when the major releases are tested because new test cases and other associated artifacts accumulate over time. The proposed approach is evaluated using widely used open source web applications with three versions.

3. REGRESSION TESTING TECHNQUIES

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of existing features. These modifications may cause the system to work incorrectly. Therefore, Regression Testing becomes necessary.

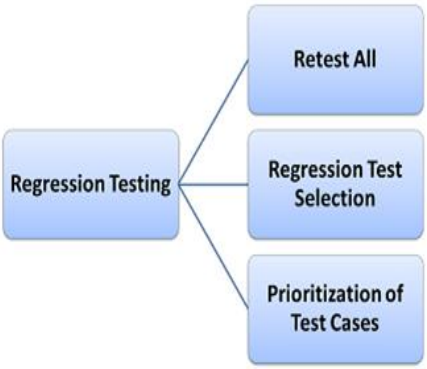

Fig 2: Regression Testing Techniques [18]

Regression Testing can be carried out using following techniques [18].

Retest All

This is one of the methods for regression testing in which all the tests in the existing test bucket or suite should be reexecuted.

- This is very expensive as it requires huge time and resources due to execution of unnecessary tests.
- When the change to a system is minor, this strategy would be rather wasteful.

Regression Test Selection

- Instead of re-executing the entire test suite, it is better to select a part of test suite to be run.
- Test cases selected can be categorized as
  a) Reusable Test Cases b) Obsolete Test Cases.

a) Re-usable Test cases can be used in succeeding regression cycles.

b) Obsolete Test Cases cannot be used in succeeding cycles.

Prioritization of Test Cases

- Prioritize the test cases depending on business impact, critical & frequently used functionalities.
- Selection of test cases based on priority will greatly reduce the regression test suite.

4. TEST CASE

Test case is the triplet [*I*,*S*,*O*], where *I* is the data input to the system, *S* is the state of the system at which the data is input, and *O* is the expected output of the system. Test suite is the set of all test cases with which a given software product is to be tested[5]. Test Cases represent a set of conditions or variables in which a tester will determine whether a system under test satisfies requirements or works correctly. Test cases should be written by a team member who understands the function or technology being tested and each test case should be submitted for peer review. Test cases act like a starting point of the test execution. Test case can be designed using only the functional specification of the software or thorough knowledge about the internal structure of software.

4.1 Types of Test Cases

There are two types of test case:

A. Formal test cases

Formal test case is designed by a known input and expected output. The known input should test a precondition and the expected output should test a post condition. If the output is as expected then the code is working correctly otherwise it needs modification. In order to fully test that all the requirements of an application are met, there must be at least two test cases for each requirement: one positive test and one negative test. The link between the requirement and the test is frequently done using a traceability matrix. Positive test will represent those conditions in which the software is working properly & negative test will specify those conditions in which the software is not working as expected or fails.

B. Informal test cases

Test cases are not written in the formal way but the activities and results are reported after the tests have been run. Generally, hypothetical stories are used to help the tester think through a complex problem or system. These scenarios are usually not written down in any detail. They can be as simple as a diagram for a testing environment or a description written in prose. Informal test case can be used for applications or systems when there is no formal requirements, test cases can be written based on the accepted normal operation of programs of a similar class.

5. TEST CASE PRIORITIZATION TECHNIQUES

Test case prioritization techniques provide a way to schedule and run test cases which have the highest priority in order to provide earlier detection of faults. Each test case is assigned a priority. Priority is set according to some criterion and test cases with highest priority are scheduled first.

For example criterion may be that the test case which has faster code coverage gets the highest priority. There are various prioritization techniques.

Some common test-case prioritization Techniques are:

a) Fault Severity

Test case prioritization depends upon the severity of faults. The order of execution of test case depends on the size of the test suite and how long each test case takes time to run. Thus, through the use of an effective prioritization technique, testers can re-order the test cases to obtain an increased rate of fault detection.

The rate of fault detection is good if

- It reveal faults earlier that have high risk
- It reveal faults related to critical code sections earlier
- Provides confidence in the system's reliability earlier

b) Code Coverage Technique

Test coverage analysis is a measure used in software testing known as code coverage analysis for practitioners. It describes the quantity of source code of a program that has been exercised during testing[19]. The following lists a process of coverage-based techniques:

- Finding areas of a program not exercised by a set of test cases
- Creating additional test cases to increase coverage
- Determining a quantitative measure of code coverage, which is an indirect measure of quality
- Identifying redundant test cases that do not increase coverage.

c) Mutation Faults

Test cases are prioritized by FEP (Fault-Exposing-Potential) Technique. This technique is achieved by the ability to expose faults and mutation analysis is used to determine this value. Each application of a mutation operator will create a mutant of the source code, which makes a single valid syntactic change. The mutation score represents the ratio of mutants that a test-suite can distinguish from the source code. The mutation score can be calculated for each test case separately[15].

FEP is calculated as

$$FEP(t_i, s_j) = \frac{|\text{mutants } (s_j) \text{ killed by } t_i|}{|\text{mutants } (s_j)|}$$

$$FEP(t_i) = \sum_j FEP(t_i, s_j)$$

Given program *P* and test suite *T*, for each test case $t \in T$ for each statement *s* in *P*, determine the mutation score of *t* on *s* i.e. the ratio of mutants of *s* exposed by *t* to total mutants of *s*. A mutant is killed by a test case if the original and the mutated program give different outputs when the test case is executed.

d) Customer Requirement-Based Prioritization

Customer requirement-based techniques are the methods to prioritize test cases based on the requirement documents. Many weight factors have been used in these techniques, including custom-priority, requirement complexity and requirement volatility[20]. Prioritization techniques use several factors to weight (or rank) the test cases. Those factors may be customer-assigned priority (CP), requirements complexity (RC) and requirements volatility (RV). The scaling factor can be assigned value (1 to 10) for each factor for the measurement. Highest factor values indicate a need for prioritization of test case related to that requirement. If requirements have high complexity then it leads to maximum number of faults.

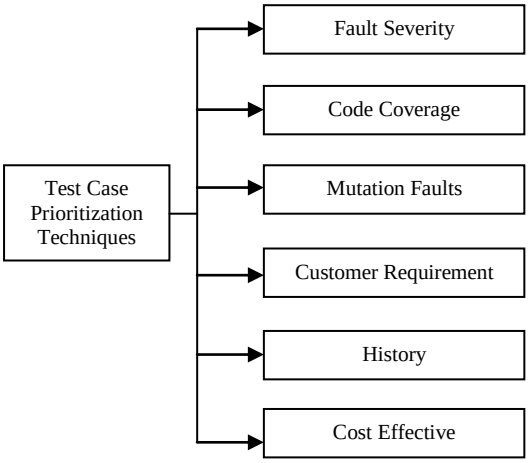

Fig 3: Test Case Prioritization Techniques

e) History based Technique

History-based techniques are the methods to prioritize test cases based on test execution history. It can enhance the effectiveness of regression testing. For a test case, using historical information of each prioritization, increase or decrease its likelihood in the current test session. A probability value is given to the history and the value of the current session is calculated based on the number of the fault detection (or coverage).

Finally, the calculated value of a test case is equal to the sum of the current number multiplied by a probability and the historical value multiplied by another probability[21].

f) Cost Effective-based Technique

Cost effective-based techniques are methods to prioritize test cases based on costs, such as cost of analysis and cost of prioritization[22]. The cost of a test case is related to the resources required to execute and validate it. Cost-cognizant prioritization requires an estimate of the severity of each fault that can be revealed by a test case. To minimize the costs associated with testing and with software failures, a goal of testing must be to uncover as many defects as possible with as little testing as possible i.e. write test cases that have a high likelihood of uncovering the faults that are the most likely to be observed as a failure in normal use. The cost–benefits should be considered first before they are applied to large programs.

Table 1. Cost Factors

Factor	Description
Execution cost	Total cost of running a set of test cases.
Cost of analysis	It includes the cost of source code analysis, analysis of changes between old and new versions and collection of execution traces.
Cost of data preparation	A total cost of preparing all input values for test cases.
Cost of validation	A total cost for validating the expected result and actual result.

6. SELECTING TEST CASES FOR REGRESSION TESTING

It was found from the industry data that good number of the defects reported by customers were due to last minute bug fixes creating side effects and hence selecting the test case for regression testing is an art and not that easy. It requires knowledge on the bug fixes and how it affect the system.

Effective Regression Tests can be done by selecting following test cases

- Test cases which have frequent defects
- Functionalities which are more visible to the users
- Test cases which verify core features of the product which are mandatory requirements of the customer
- Test cases of functionalities which has undergone more and recent changes
- All Integration Test Cases
- All Complex Test Cases
- Boundary value test cases
- Sample of Successful test cases
- Sample of Failure test cases

Selection of test cases for regression testing depends more on the criticality of bug fixes than the criticality of the defect itself. A minor defect can result in major side effect and a bug fix for an extreme defect can have no or a just a minor side effect. So the test engineer needs to balance these aspects for selecting the test cases for regression testing.

7. REGRESSION TESTING TOOLS

When a software undergoes frequent changes, regression testing costs will escalate. In such cases, manual execution of test cases increases test execution time as well as costs. Automation of regression test cases is required in such cases. Extent of automation depends on the number of test cases that remain re-usable for successive regression cycles.

Following are most important tools used for both functional and regression testing[18]:

*Quick Test Professional (QTP):*HP Quick Test Professional is automated software designed to automate functional and regression test cases. It uses VbScript language for automation. It is a Data driven , Keyword based tool.

*Rational Functional Tester (RFT):*IBM's rational functional tester is a java tool used to automate the test cases of software applications. This is primarily used for automating regression test cases and it also integrates with Rational Test Manager.

Selenium: This is an open source tool used for automating web applications. Selenium can be used for browser based regression testing.

8. CONCLUSION

In this paper, regression testing and test case prioritization methodologies are presented. There are various factors on the basis of which test case prioritize can be decided. It includes customer requirements, history based, cost analysis, fault exposing, code coverage etc. Test prioritization can strengthen regression testing for finding more severe fault in earlier stages. Test case prioritization varies from project to project. With earlier prioritization of test cases we can reduce cost, time, effort and maximize customer satisfaction. A system can be developed using MATLAB to analyze the test case prioritization after performing regression testing on the developed software. In future, if we have a large test suite then we can implement the clustering to categorize the faults and then perform the cluster based prioritization approach. Moreover for these techniques, soft computing approaches like Genetic Algorithms, Fuzzy Logic, Artificial Neural Network etc. may be used for experimentation and validation purpose.

9. REFERENCES

[1] Pressman, Roger S., "Software engineering: A practitioner's approach", McGraw-Hill Companies, 5th edition, 2005.

[2] Sommerville, Ian. "Software Engineering", Addison Wesley, 9th edition ,2011.

[3] Hema Srikanth, Laurie Williams and Jason Osborne, “System Test Case Prioritization of New and Regression Test Cases”, Proceedings of the 4th International Symposium on Empirical Software Engineering (ISESE), pp.62–71, IEEE Computer Society, 2005.

[4] Dennis Jeffrey and Neelam Gupta, “Test Case Prioritization Using Relevant Slices”, Proceedings of the 30th Annual International Computer Software and Applications Conference, Volume 01, 2006, pp.411-420, 2006.

[5] Mall, Rajib, "Fundamentals of Software engineering", PHI Learning Pvt. Ltd., 2014.

[6] http://www.onestoptesting.com/regression-testing/types.asp

[7] Prakash, N., & Rangaswamy, T. R., "Modular based multiple test case prioritization", Computational Intelligence & Computing Research (ICCIC), IEEE International Conference (pp. 1-7),2012

[8] Raju, S., and G. V. Uma. "Factors oriented test case prioritization technique in regression testing using genetic algorithm." European Journal of Scientific Research ,pp. 389-402,2012.

[9] Chen, L., Wang, Z., Xu, L., Lu, H., & Xu, B. "Test case prioritization for web service regression testing",Service Oriented System Engineering (SOSE), Fifth IEEE International Symposium (pp. 173-178). 2010.

[10] Duggal, Gaurav, and Bharati Suri. "Understanding regression testing techniques." Proceedings of the 2nd National Conference on Challenges and Opportunities, Mandi Gobindgarh, India, March. Vol. 29. 2008.

[11] Joshi, Shweta A., and B. S. Tiple. "Literature Review of Model Based Test case Prioritization." International Journal of Computer Science & Information Technologies,Volume 5, no. 5 ,2014.

[12] Roongruangsuwan, Siripong. Jirapun Daengdej,"Test Case prioritization techniques." Journal of theoretical and applied informational technology,2005.

[13] Chaturvedi, Animesh, and Atul Gupta. "A tool supported approach to perform efficient regression testing of web services." Maintenance and Evolution of Service-Oriented and Cloud-Based Systems (MESOCA), IEEE 7th International Symposium, pp. 50-55, 2013.

[14] Mei, L., Zhang, Z., Chan, W. K., & Tse, T. H. ,"Test case prioritization for regression testing of service-oriented business applications", Proceedings of the 18th international conference on World wide web, pp. 901-910. ACM,2009.

[15] Elbaum, S., Malishevsky, A. G., & Rothermel, G.,"Prioritizing test cases for regression testing",ACM,Vol. 25, No. 5, pp. 102-112,2000

[16] Lin, C. T., Chen, C. D., Tsai, C. S., & Kapfhammer, G. M.,"History-based test case prioritization with software version awareness", 18th International Conference on Engineering of Complex Computer Systems (ICECCS), pp. 171-172, 2013.

[17] Md. Hossain, Hyunsook Do, Ravi Eda," Regression Testing for Web Applications Using Reusable Constraint Values", IEEE International Conference on Software Testing, Verification, and Validation Workshops,pp-312-321 ,DOI 10.1109/ICSTW.2014.35,2014

[18] http://www.guru99.com/regression-testing.html

[19] 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.

[20] G. Rothermel, R. Untch, C. Chu, and M. Harrold,"Test Case Prioritization", IEEE Transactions on Software Engineering, vol. 27, pp. 929-948, 2001.

[21] Jung-Min Kim and A. Porter.,"A history-based test prioritization technique for regression testing in resource constrained environments", *ICSE '02:* Proceedings of the 24th International Conference on Software Engineering, pages 119–129, New York, NY, USA, 2002. ACM Press.

[22] Alexey G. Malishevsky, Gregg Rothermel and Sebastian Elbaum, "Modeling the Cost-Benefits Tradeoffs for Regression Testing Techniques", Proceedings of the International Conference on Software Maintenance (ICSM'02), 2002.