Volume 8, No. 3, March – April 2017
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Time & Quality Improvement of Regression Testing using Pruning Method

Kanika Sharma and Amit Jain
Research Scholar and Associate Professor
Department of Computer Science and Technology
Ludhiana college of Engineering and technology
Ludhiana, Punjab, India

Abstract- Software tests, specifically software represented through regression testing, it accompanies the entire lifestyles cycle of industrial software machine. In this report a regression testing technique for enterprise-orientated programs to solve troubles such as time. Regression Testing is a form of software program testing that verifies that software previously evolved and tested nevertheless performs efficiently even after it turned into modified or interfaced with different software program. Changes may consist of software program enhancements, patches, configuration modifications, and many others. At some stage in regression trying out, new software program insects or regressions may be uncovered. Sometimes a software trade effect analysis is done to decide what regions could be suffering from the proposed modifications. Those areas may encompass purposeful and non-functional regions of the gadget. In this proposed work, pruning of regression testinghas been performed by designing a mathematical model.For that, five different parametersand two test-case scenarios have been chosen and the effect of individual change or combinational changes in parameters have been analyzed. The automated pruning on the basis of changes in these parameters has been performed.The parameters have been chosen in which changes are made and then scenarios of change have been selected to generate various tests cases. The results showed that the objectives have been fully achieved after pruning the results and has improved both quantitatively and qualitatively in terms of their cost, risk and time factors. The time taken for testing has been minimized up to 50% whereas cost and risk factors has also been improved very effectively when compared with the results obtained from existing technique.

Keywords: Regression testing, industry applications, change impact analysis.

I. INTRODUCTION

Software Testing: Software testing is the concept which verifies that the software given to end user is upto standards. The standard of software testing is maintained by auditing the software products and reviewing its development activities. Software testing process will be greatly promoted by popularizing the test model specification and quality of the software will also be improved. However, In traditional software test models the focused attention is laid on to software development testing process, they did not emphasis to regression testing, and it would not solve the problem of accumulating knowledge in the industrial software testing application [8]. With the wide use of IT in different businesses, software systems have turned out to be extraordinarily necessary. The unwavering quality of the system is taking part in a key supporting part within the application business advancement. Advancement and upkeep are constantly gone with the whole lifecycle of application systems. Subsequently, there is a developing interest for regression testing.

1.2 Regression Testing:

Regression trying out is described as a kind of software program trying out to affirm that a recent program or code alternate has no longer adversely affected existing functions. Regression testing is not anything but complete or partial choice of already done take a look at cases that are re-executed to make certain current functionalities work fine. This testing is carried out to make certain that new code changes need to no longer have aspect outcomes on the prevailing functionalities. It guarantees that old code nevertheless works as soon as the brand new code modifications are executed.

Need of Regression Testing

Regression Testing is required when there may be a

- Change in requirements and the code is changed according to the requirement
- The new feature is brought to the software program
- Defect fixing
- Performance problem fixes

Regression Testing Techniques

Software upkeep is an activity which incorporates improvements, error corrections, optimization and deletion of existing functions. These changes may motive the gadget to work incorrectly. Therefore, Regression Testing will become essential. Regression Testing can be achieved the use of the following strategies [9]:

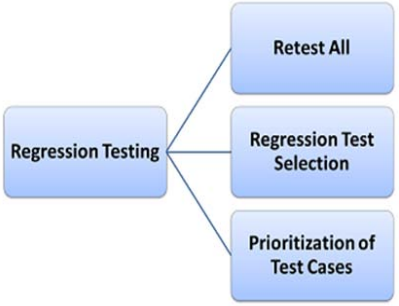


Fig 1.1 Regression tests techniques

1.3 Cost risk assessment

Assessing risk way, figuring out the effects (including fees) of capability dangers. Risk tests include asking questions inclusive of: Is this a risk or no longer? How severe is the hazard? What are the results? What is the likelihood of this risk occurring? Decisions are made based on the chance being assessed. The choice(s) can be to mitigate, manipulate or forget about. The vital matters to identify (and quantify) are:

- What signs can be used to be expecting the chance of a failure?
- The vital issue is to pick out what's important to the pleasantness of this function. This may additionally include design quality (e.g. What number of change requests needed to be raised), program size, complexity, programmer's competencies etc.
- What are the outcomes if this specific function fails?
- Very frequently is it impossible to quantify this accurately, but the use of low-medium-high (1-2-three) may be appropriate enough to rank the individual functions.
- By combining the consequence and the possibility (from threat identity above) it needs to now be feasible to rank the individual capabilities of a machine. The ranking could be finished based totally on "experience" or by empirical calculations.

1.4 Change Impact Analysis

Impact analysis is a key thing of accountable necessities management. It provides the accurate know-how of the results of a proposed change, which facilitates the crew make informed commercial enterprise selections approximately which proposals to approve. The analysis examines the proposed change to become aware of additives that could have to be created, modified, or discarded and to estimate the effort associated with imposing the alternate. Skipping impact evaluation doesn't change the scale of the mission. It simply turns the dimensions into a surprise. Software surprises are rarely appropriate news. Before a developer says, "Sure, no problem" in response to an alternate request, he or she have to spend a touch time on impact evaluation.

Impact Analysis Procedure

The chairperson of the exchange manages board will usually ask an informed developer to perform the impact analysis for a particular alternate inspiration. Impact evaluation has 3 aspects:

1. Understand the feasible implications of creating the trade. Change often produces a huge ripple impact. Stuffing too much functionality into a product can lessen its performance to unacceptable stages, as when a system that runs each day calls for extra than 24 hours to complete an unmarried execution.
2. Identify all the files, fashions, and files that would need to be modified if the group incorporates the requested alternate.
3. Identify the obligations required to put into effect the alternate, and estimate the attempt wanted to finish those obligations [10].

1.5 Business Rule

A business rule is a server-side script that runs when a report is displayed, inserted, up to date, or deleted, or while a desk is queried. Use business policies to accomplish tasks like mechanically changing values in shape fields whilst positive conditions are met, or to create events for email notifications and script actions [11].

A business rule defines or constrains one issue of your business that is supposed to claim business shape or impact the conduct of your commercial enterprise. Business guidelines regularly cognizance on get entry to manage problems, as an instance, professors are allowed to import and modify the marks of the students taking the seminars they instruct, but not the marks of college students in other seminars. Business rules may additionally pertain to enterprise calculations, as an example, a way to convert a percentage mark (as an example, 91 percent) that a pupil gets in a seminar right into a letter grade (for instance, A-) [12].

II. LITERATURE REVIEW

Yanlin Li et.al in [1] proposed a way to deal with investigate the attainable execution heading of programming project segments disappeared with refresh display basically in light of use lessening. Some coordinating example models are suggested and they could incrementally make greater the experiments that neglect to fit. The structure-orientated relapse check ways might be without issues procured from these coordinating example paradigms. In addition, the computerized time of relapse test way can be completed inside the procedure of programming system item development. An occurrence is additionally exhibited to demonstrate the era of relapse check course inside the level of source codes.

Milos Gligoric et.al in [2] portrayed a relapse test-determination method for programming created utilizing current disseminated rendition control frameworks. By demonstrating distinctive branch or union summons specifically in their system, it registers safe test sets that can be generously littler than applying past methods to a linearization of the product history. They assess their system on programming histories of a few expansive open-source ventures. The outcomes are empowering: their procedure got a normal of 10.89× diminishment in the quantity of tests over a current strategy while as yet choosing all tests whose conduct may contrast.

Sascha Lity et.al in [3] proposed programmed substitute impact examination in view of incremental model diminishing for incremental SPL looking at. Incremental cutting grants for a cut calculation by methods for adjusting a past cut with particular deduction in their varieties by utilizing considering form alterations. They rehearse incremental slicing to decide the impact of completed form changes and to reason around their ability retest. In view of their novel retest protection model, each cut trade indicates a retest check goal to be secured by means of present investigate cases chose for retesting. They prototypically connected their approach and assessed its appropriateness and adequacy by utilizing 4 SPLs.

Hyunsook Do et.al in [4] proposed another relapse looking at system that distinguishes the influenced zones through code changes utilizing sway assessment and creates new check cases for the affected districts by means of changes utilizing application cuts. To encourage the approach, the analysts completed a (PHP) Analysis and Regression Testing Engine (PARTE) and finished an oversaw explore the utilization of 5 open source web programs with several varieties. The impacts affirmed that this approach is compelling in diminishing the expense of relapse looking at for a consistently fixed web utility, and uncovered methodologies wherein that viability can run with application qualities and forming frequencies.

Francisco Zapata et.al in [5] proposed the utilization of Basis Path Testing, that is a white-field programming testing strategy that utilizations Graph Theory to look at the intricacy of a needy machine by methods for making a control accept the way things are chart from everything about framework's capacities to plan a debut test suite. This test suite is a settled of ways that navigate by means of the capacities, which may be accepted straightly unbiased and that can be utilized to make a test procedure so one can work out the majority of the product's capacities at any rate when to affirm and approve their usefulness. By making utilization of Basis Path Testing examination to the constituent structures in a SoS, the analyzer can grow a most dependable check suite with a view to ensure that everybody possible fair ways, all reasonable intelligent choices, and every one of their interfaces are executed as a base once. This paper manages a SoS test structure and show how to produce a test suite the utilization of Basis Path Testing examination.

Bernhard K. Aichernig et.al in [6] exhibited the procedures and impacts of a novel form based investigate case innovation strategy that routinely gets test occasions from UML country machines. The essential commitment of this article is the totally programmed blame based thoroughly check case innovation approach together with two observational contextual investigations got from business utilize cases. Likewise, a top to bottom appraisal of various blame essentially construct experiment innovation procedures in light of everything about contextual investigations is given and an evaluation with verifiable arbitrary experimenting with is directed. The experiment period technique underpins an extensive variety of UML builds and is grounded at the formal semantics of Back's movement structures and the outstanding enter–output conformance connection. Transformation administrators are utilized on the degree of the particular to embed blames and produce investigate examples with a reason to screen the shortcomings embedded. The adequacy of this approach is appeared and it is talked about the best approach to pick up a more prominent expressive investigate suite by methods for joining shabby yet undirected arbitrary investigate case time with the additional expensive however coordinated transformation based absolutely technique. At last, an inside and out and vital dialog of the preparation learnt is given notwithstanding a fate point of view toward the general helpfulness and practicability of transformation based thoroughly experiment period.

Mustafa Al-Hajjaji et.al in [7] proposed strategy does now not ensure to discover a bigger number of errors than examining systems, in any case it goes for developing exchange protection of a SPL underneath investigate quick as practical throughout the years. This is especially valuable considering that for the most part the time spending plan for testing is limited. They executed likeness essentially based prioritization in Feature IDE and assessed it by method for assessing its result to the default last aftereffects of 3 testing calculations notwithstanding irregular requests. The test outcomes recommend that the request with similitude based prioritization is superior to anything arbitrary requests and much of the time superior to anything the default request of current examining calculations.

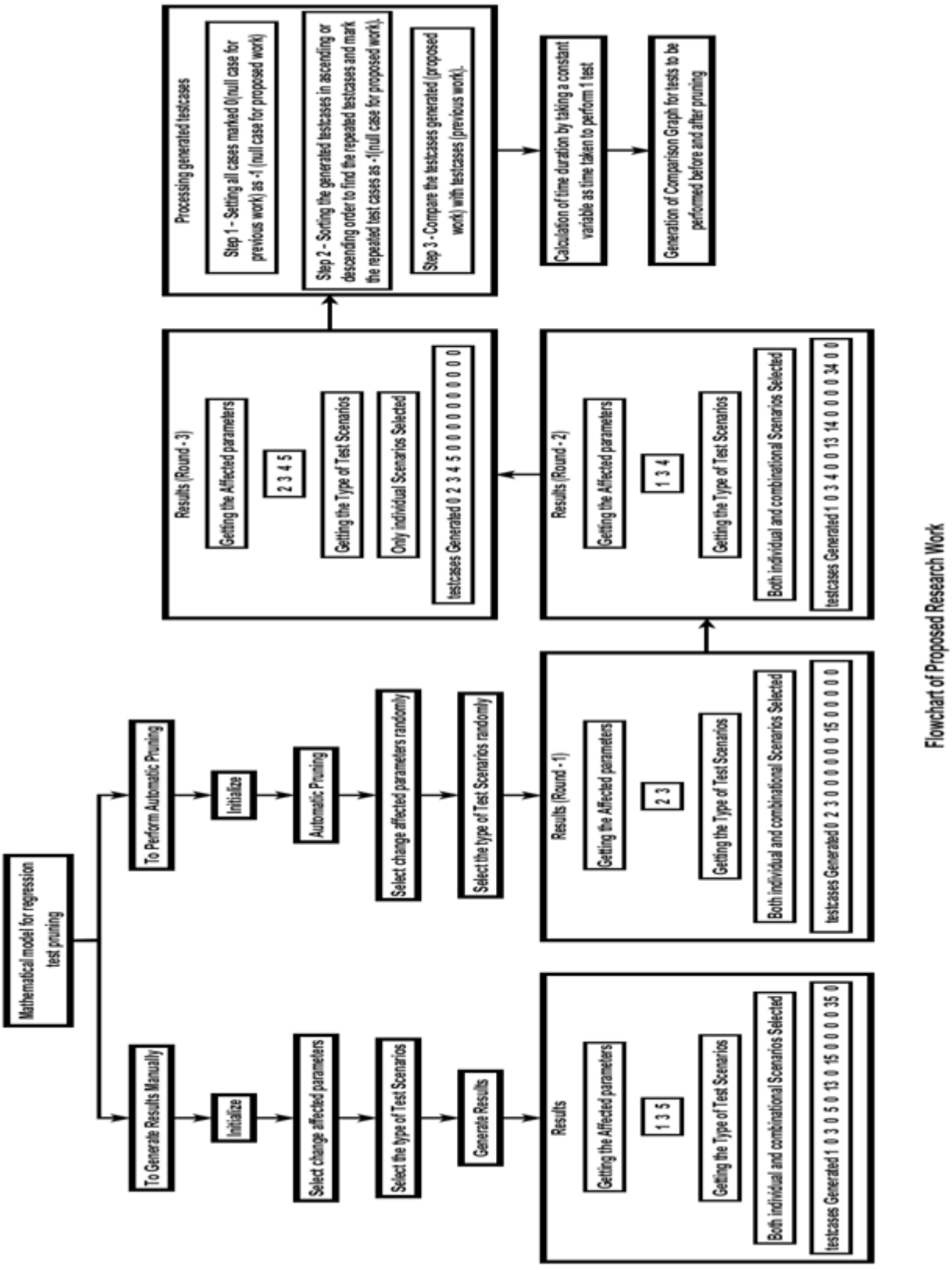
III. PROBLEM FORMULATION

Regression testing is the way toward testing changes to computer programs to ensure that the more seasoned programming still works with the new alterations. Regression testing is an ordinary piece of the program improvement process and, in bigger organizations, is finished by code testing specialists. Test department coders create code test situations and activities that will test new units of code after they have been composed. These experiments or test cases or scenarios built what turns into the test bucket. Before a new build of a software product is released, the old experiments are keeping running against the new form to ensure that all the old capacities still work. The reason they won't not work is on account of changing or adding new code to a program can undoubtedly bring blunders into code that is not expected to be changed.

The need and handwork of modifying the base system (Regression Testing) is felt because the tests which are non - pruned take a lot of time and effort each time the system has to be applied. The need of a system is felt which can be test system according to need i.e. the only part of code should be tested in which changes are made and where the impact of changes is possible; there is no need to test the whole software

again and again. This type of testing is called pruned testing in which useful pinpoint test which can focuses on the set of test required are performed. The system has to be pruned for more and more useful tests.

IV. METHODOLOGY



Flowchart of Proposed Research Work

V. RESULTS

Regression implies retesting the unaltered parts of the application. Test cases are re-executed so as to check whether past utilization of software application is working fine and new changes have not presented any new bugs. This test can be performed on a new version when there is noteworthy change in original practicality or even a single bug settle. This is the strategy of confirmation. Checking that the bugs are settled and the recently included features have not made any issue in past working edition of software. The need of a system is felt which can be test system according to need i.e. the only part of code should be tested in which changes are made and where the impact of changes is possible; there is no need to test the whole software again and again. This type of testing is called pruned testing in which useful pinpoint test which can focuses on the set of test required are performed. The system has to be pruned for more and more useful tests.

We proposed to design and perform pruning of regression testing. We have designed a mathematical model for regression test pruning. We have chosen 5 parameters and two tests-case scenarios as the effect of changes in parameters may be individual change or combinational change. We performed automated pruning on the basis of changes in these parameters, the parameters are chosen in which changes are made and then scenarios of change are selected (i.e. Individual or Combinational or Both or No Scenario) to generate tests to be performed.

Table 7.1 – Showing Parameters and Test Scenarios

Parameters	Test Scenario Individual	Test Scenario Combinational
Parameter P1	Parameter P1	Parameter P1, Parameter P2
Parameter P2	Parameter P2	Parameter P1, Parameter P3
Parameter P3	Parameter P3	Parameter P1, Parameter P4
Parameter P4	Parameter P4	Parameter P1, Parameter P5
Parameter P5	Parameter P5	Parameter P2, Parameter P3
-	-	Parameter P2, Parameter P4
-	-	Parameter P2, Parameter P5
-	-	Parameter P3, Parameter P4
-	-	Parameter P3, Parameter P5
-	-	Parameter P4, Parameter P5

Steps for Simulation: -

A. Generate Results Manually

- Step 1 – Press Initialize Button.
- Step 2 – Select the parameters in which changes are made.
- Step 3 – Select the type of test scenario (i.e. Individual or Combinational or Both or No Scenario) to generate tests to be performed.
- Step 4 – Press Generate Results' button.
- Step 5 – See Results Window for generating test cases.
  - Getting the Affected parameters
  - Showing Parameters Affected.
  - Getting the Type of Test Scenarios
  - Showing the type of Test Scenarios selected Individual or Combinational or Both or No Scenario.
  - Showing testcases Generated for testing purpose.

B. Perform Automatic Pruning

In automated pruning parameter selection and scenario selection are done randomly as we are performing automated pruning, but in actual real conditions we have to give manual input of parameters in which changes are made and what type of test scenarios we like to perform.

- Step 1 – Press Initialize Button.
- Step 2 – Press Automated Pruning Button.
- Step 3 – See Results Window for generating test cases.
  - Getting the Affected parameters
  - Showing Parameters Affected.
  - Getting the Type of Test Scenarios
  - Showing the type of Test Scenarios selected Individual or Combinational or Both or No Scenario.
  - Showing testcases Generated for testing purpose.

In automated pruning three rounds of testing are made to be sure for results generated.

1. Generate Results Manually

Fig 7.1 – Mathematical model for regression test pruning

Figure 7.1 shows a mathematical model for regression test pruning. A simulation screen showing the initialize button to reset the system, Select parameters panel in which different

buttons are given to select parameters and test scenario selection panel to select a test scenario like Individual or Combinational or Both or No Scenario and two buttons to perform automated pruning or to generate results manually and lastly its showing results panel in which results generated are displayed.

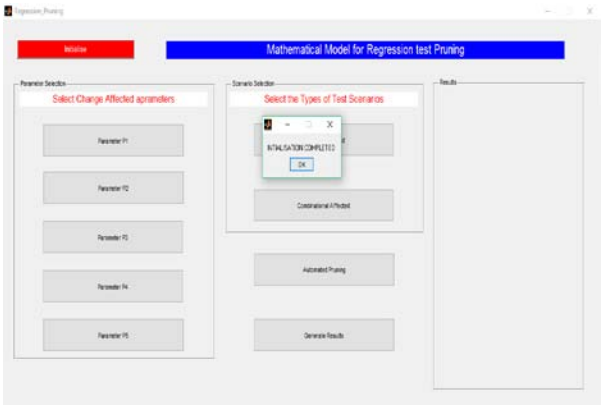


Fig 7.2 – Initialization Message

Figure 7.2 shows the message initialization completed. In simulation this message pops up when we press the initialize button. Before starting any testing or pruning we have to initialize the system. When we press initialization button it resets the value of all the variables, parameters, and scenarios. It also clears any previous results generated showing in the results panel. Please press on "OK" or cross sign on the message box to proceed to next step selection of change affected parameters.

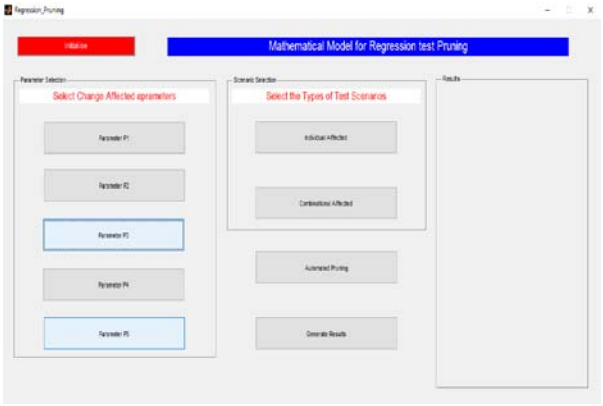


Fig 7.3 – selection of parameters and test scenarios

Figure 7.3 shows the process after initialization, now we have to choose change affected parameters. Here we are considering that we are having 5 parameters named P1, P2, P3, P4 and P5. We can choose one or more parameter according to changes in parameters. After parameter selection we have to choose the type of test scenarios we want to test as changes may affect the parameters individually or combination ally.

Fig 7.4 – Command Window shows selected parameters

Figure 7.4 shows that command window is shown here we have selected 3 parameters P1, P3 and P5.

Fig 7.5 – Generated results

Figure 7.5 shows generate results' button is pressed. Result panel is displaying results according to parameters selected and test scenarios selected. Result panel showing: -

- Getting the Affected parameters 1 3 5
- Getting the Type of Test Scenarios
- Both individual and combinational Scenarios Selected
- Testcases Generated 1 0 3 0 5 0 13 0 15 0 0 0 0 35 0

2. Perform Automatic Pruning

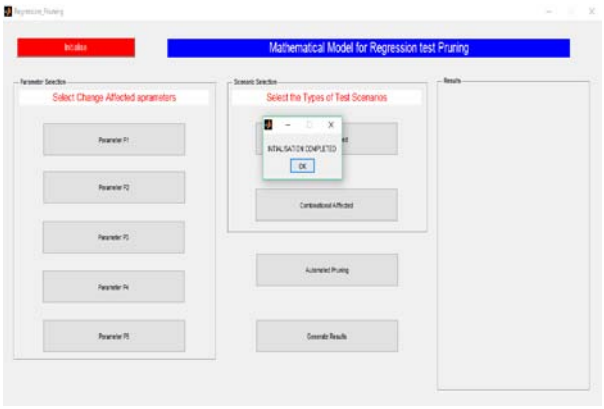


Fig 7.6 – Initialization Message

Figure 7.6 shows the message initialization completed. In simulation this message pops up when we press the initialize button. Before starting any testing or pruning we have to initialize the system. When we press initialization button it

resets the value of all the variables, parameters, and scenarios. It also clears any previous results generated showing in the results panel. Please press on "OK" or cross sign on the message box to proceed to next step perform automated pruning.

Fig 7.7 – Perform Automated Pruning

Figure 7.7 shows after initialization of the system we are going to perform automated pruning. In automated pruning parameter selection and scenario selection are done randomly as we are performing automated pruning, but in actual real conditions we have to give manual input of parameters in which changes are made and what type of test scenarios we like to perform. We press the button automated pruning and the results are displayed in the results panel.

In automated pruning three rounds of testing are made to be sure for results generated.

Fig 7.8 – Automated Pruning results generated round 1

Figure 7.8 shows Automated Pruning results generated round 1. Result panel showing: -

- Getting the Affected parameters 2 3
- Getting the Type of Test Scenarios
- Both individual and combinational Scenarios Selected
- Testcases Generated 0 2 3 0 0 0 0 0 0 23 0 0 0 0 0

Fig 7.9 – Automated Pruning results generated round 2

Figure 7.9 shows Automated Pruning results generated round 2. Result panel showing: -

- Getting the Affected parameters 1 3 4
- Getting the Type of Test Scenarios
- Both individual and combinational Scenarios Selected
- Testcases Generated 1 0 3 4 0 0 13 14 0 0 0 0 34 0 0

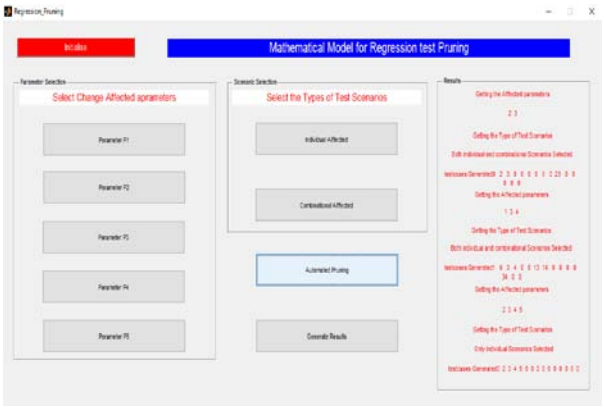


Fig 7.10 – Automated Pruning results generated round 3

Figure 7.10 shows Automated Pruning results generated round 3. Result panel showing: -

- Getting the Affected parameters 2 3 4 5
- Getting the Type of Test Scenarios
- Only individual Scenarios Selected
- Testcases Generated 0 2 3 4 5 0 0 0 0 0 0 0 0 0 0

In automated pruning three rounds of testing are made to be sure for results generated. These automated pruning results are then compared with results before pruning regression test results. In normal scenario we find that we are having total 15 test cases which are to be performed combining individual and combinational testcases as mentioned in Table 7.1. By performing three rounds of test we find many test cases are repeating therefore we have to omit repeated testcases. Therefore, we are processing results generated. The steps of processing generated test cases are as follows:-

Step 1 – Setting all cases marked 0 (null case for previous work) as -1 (null case for proposed work) so that while making comparison, these cases not match with cases in normal regression test cases as in normal testing there may be cases which are marked as zero.

```
For i=1: 45
  if (testdirectory(i)==0)
  testdirectory(i)=-1;
  End
End
```

Step 2 – Sorting the generated test cases in ascending or descending order to find the repeated testcases and mark the repeated test cases as -1 (null case for proposed work).

```
testdirectory=sort(testdirectory);
For i=2: 45
if (testdirectory(i)==testdirectory(i-1))
testdirectory(i)=-1;
End
End
```

Step 3 – Compare the test cases generated (proposed work) with test cases (previous work).

```
For i=1: 15
For j=1: 45
        if (stestcase(i)==testdirectory(j))
        correct=correct+1;
        End
End
End
```

All the cases find correct with previous work for testing are the final test cases which are to be performed.

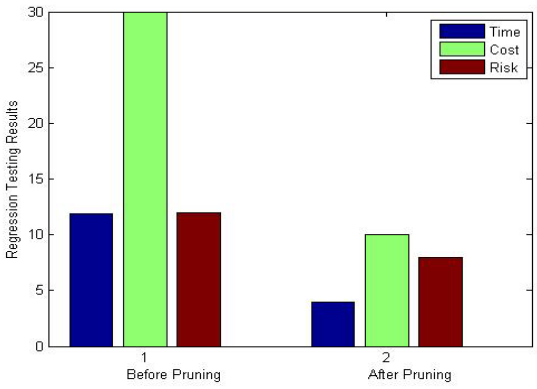


Fig 7.11 – Graph shows various parameters for regression testing with a proposed method before and after pruning

Figure 7.11 Graph shows various parameters, i.e. time, cost and risk for regression testing with a proposed method before and after pruning. The graph is plotted as follows a variable is taken time cycle in which random value is generated. Suppose the value generated is 1 so the number of testcases multiplied by value of time cycle will be time taken in seconds of previous work which is stored in variable plo1 and no. of testcases minus correct test cases detected will be time taken in seconds for proposed work which is stored in variable plo2. The cost for both scenarios calculated as follows:-

Cost before pruning = Cost of 1 unit testcases * (plo1/plo2)

Cost after pruning = Cost of the 1 unit testcases * (plo2/plo2)

Cost before pruning cost will be 10*(plo1/plo2) in which 10 is the cost of 1 unit test cases to be performed and the cost after pruning cost will be 10*(plo2/plo2) in which 10 is the cost of 1 unit test cases to be performed.

The risk factor is calculated on the basis of three factors as follows:-

Factor1 = No. of Software / No. of Employees

Factor2 = Efficiency of Employees / Accuracy of Software

MaxRisk = 50

Before Pruning Risk shows:

Factor1 = 180/6 = 30

Factor2 = 90/3 = 30

Accuracy of software = 3 out of 10

Before Risk = (((Factor1+Factor2)/MaxRisk)*10) = 12 risks

After Pruning Risk shows:

Factor1 = 180/6 = 30

Factor2 = 90/9 = 10

Accuracy of software = 9 out of 10

After Risk = (((Factor1+Factor2)/MaxRisk)*10) = 8 risks

Fig 7.12 – Command Window shows values of various parameters for regression testing

Table 7.2 – Time taken for regression testing

Parameter	Before Pruning	After Pruning
Time	11.9280	3.9760
Cost	30	10
Risk	12	8

Figure 7.12 shows the Command Window shows time, cost and risk taken for regression testing. Time Taken for Regression Testing (in Seconds) Before Pruning is 11.9280 and After Pruning is 3.9760. Cost for Regression Testing (in units) Before Pruning is 30 and After Pruning is 10. Risk for Regression Testing (in Percentage) Before Pruning is 12 and After Pruning is 8.

Fig 7.13 – Message box Please Press Initialize Button

Figure 7.13 shows the message box Please Press Initialize Button. This message box pop ups when we press any parameter selection button or result generation button before initializing the system.

VI. CONCLUSION AND FUTURE WORK

Regression testing is going with the entire procedure life cycle of the software application system. During the entire procedure, it will produce a tons of test reports, which contains a lot of information and experience. We have designed and performed pruning of regression testing. We have designed a mathematical model for regression test pruning. We have chosen 5 parameters and two tests-case scenarios as the effect of changes in parameters may be individual change or combinational change. We performed automated pruning on the basis of changes in these parameters, the parameters are chosen in which changes are made and then scenarios of change are selected (i.e. Individual or Combinational or Both or No Scenario) to generate test cases to be performed. The results show that our objectives are achieved after pruning the results are improved comparatively and time taken for testing is minimized up to 50% whereas cost and risk factors are also improved effectiveness.

The future scope of the work is to design a smart system with the automatic sequence recognition system. The system will automatically recognize the pattern of changes and will perform smart testing accordingly with respect to the changed parameters.

REFERENCES

[1] Li, Y., Du, J., Hu, Q. and Liu, X., 2016, October. A Method for Structure-Oriented Regression Test Path Generation. In System and Software Reliability (ISSSR), International Symposium on (pp. 30-36). IEEE.

[2] Gligoric, M., Majumdar, R., Sharma, R., Eloussi, L. and Marinov, D., 2014, July. Regression test selection for distributed software histories. In International Conference on Computer Aided Verification (pp. 293-309). Springer International Publishing.

[3] Lity, S., Morbach, T., Thüm, T. and Schaefer, I., 2016, June. Applying Incremental Model Slicing to Product-Line Regression Testing. In International Conference on Software Reuse (pp. 3-19). Springer International Publishing.

[4] Do, H. and Hossain, M., 2014. An efficient regression testing approach for PHP web applications: a controlled experiment. Software Testing, Verification and Reliability, 24(5), pp.367-385.

[5] Zapata, F., Akundi, A., Pineda, R. and Smith, E., 2013. Basis Path Analysis for Testing Complex System of Systems. Procedia Computer Science, 20, pp.256-261.

[6] Aichernig, B.K., Brandl, H., Jöbstl, E., Krenn, W., Schlick, R. and Tiran, S., 2015. Killing strategies for model-based mutation testing. Software Testing, Verification and Reliability, 25(8), pp.716-748.

[7] Al-Hajjaji, M., Thüm, T., Meinicke, J., Lochau, M. and Saake, G., 2014, September. Similarity-based prioritization in software product-line testing. In Proceedings of the 18th International Software Product Line Conference-Volume 1 (pp. 197-206). ACM.

[8] Regression Testing. 2016. Regression Testing. [ONLINE] Available at: https://www.cs.umd.edu/~aporter/html/currTesting.html. [Accessed 17 December 2016].

[9] What is Regression testing in software?. 2017. What is Regression testing in software?. [ONLINE] Available at: http://istqbexamcertification.com/what-is-regression-testing-in-software/. [Accessed 22 February 2017].

[10] Jama Software. 2017. Best Practices for Change Impact Analysis | Jama Software. [ONLINE] Available at: http://www.jamasoftware.com/blog/change-impact-analysis-2/. [Accessed 22 February 2017].

[11] ServiceNow. 2017. Business Rules - ServiceNow Wiki. [ONLINE] Available at: http://wiki.servicenow.com/index.php?title=Business_Rules#gsc.tab=0. [Accessed 22 February 2017].

[12] Business Rules: An Agile Introduction. 2017. Business Rules: An Agile Introduction. [ONLINE] Available at: http://agilemodeling.com/artifacts/businessRule.htm. [Accessed 22 February 2017].