LOVELY PROFESSIONAL UNIVERSITY

Optimization of Regression Testing Technique for Web Application

A Dissertation Submitted

By

Sarbjot Kaur

to

Department of Computer Science and Engineering

In partial fulfillment of the Requirement for the

Award of the Degree of

Master of Technology in Computer Science & Engineering

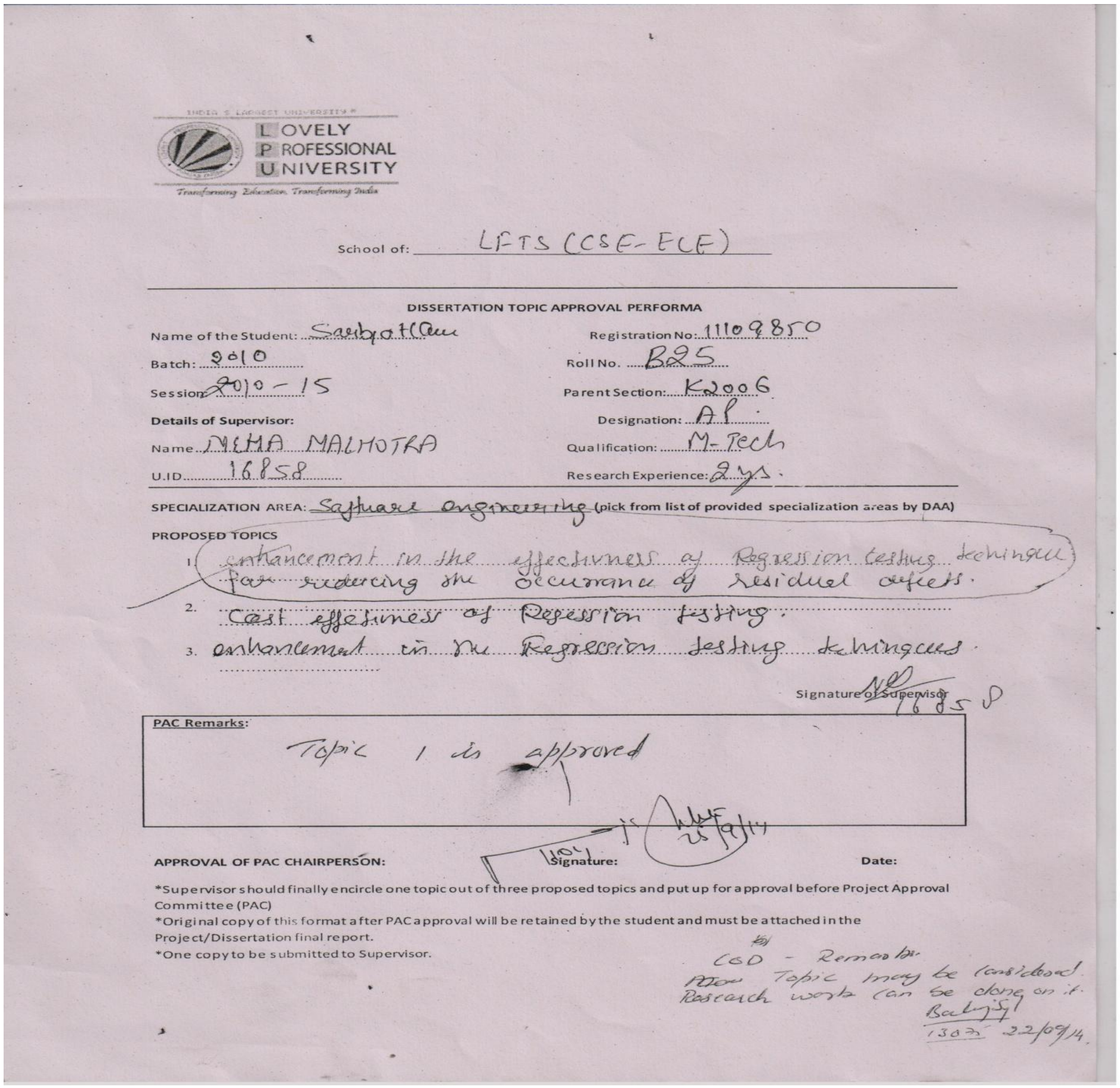
Under the guidance of

Mr. Makul Mahajan

(Assistant Professor)

(April 2015)

PAC FORM



INDIA'S LARGEST UNIVERSITY

LOVELY PROFESSIONAL UNIVERSITY

Transforming Education, Transforming India

School of: LFTS (CSE-ECE)

DISSERTATION TOPIC APPROVAL PERFORMA

Name of the Student:	Sarbjot Kaur	Registration No:	11109850
Batch:	2010	Roll No.	B25
Session:	2010 - 15	Parent Section:	K2006
Details of Supervisor:		Designation:	AP.
Name:	NEHA MALHOTRA	Qualification:	M-Tech
U.ID	16858	Research Experience:	2 yrs.

SPECIALIZATION AREA: Software engineering **(pick from list of provided specialization areas by DAA)**

PROPOSED TOPICS

1. enhancement in the effectiveness of Regression testing technique for reducing the occurrence of residual defects.
2. Cost effectiveness of Regression testing.
3. enhancement in the Regression testing techniques.

Signature of Supervisor [illegible] 16858

PAC Remarks:

Topic 1 is approved

[illegible] 25/9/14

APPROVAL OF PAC CHAIRPERSON: **Signature:** **Date:**

*Supervisor should finally encircle one topic out of three proposed topics and put up for approval before Project Approval Committee (PAC)

*Original copy of this format after PAC approval will be retained by the student and must be attached in the Project/Dissertation final report.

*One copy to be submitted to Supervisor.

COD - Remarks.
Above Topic may be considered.
Research work can be done on it.
[illegible]
13037 22/09/14.

ABSTRACT

Regression testing means re-testing an application after its code has been modified to verify that it still functions correctly. Regression testing consists of re-running existing test cases and checking that code changes did not break any previously working functions, inadvertently introduce errors or cause earlier fixed issues to reappear. Regression testing is also use to test the web application. Web testing is the name given to software testing that focuses on web applications. Complete testing of a web-based system before going live can help address issues before the system is revealed to the public. The test case used for the regression testing has been reused to save time and effort required to develop new test cases. To make the validate result chksim algorithm will be used. The test cases generated will be optimized using expert system.

ACKNOWLEDGEMENT

First and foremost I would like to thank almighty for giving me courage to bring up this dissertation. Before getting into thick and thin of this dissertation I would like to show my gratitude to some of the people who have helped me in this project. Firstly I would like to propose a word thanks to my mentor Mr.Makul Mahajan who has encouraged me to get through this dissertation. Secondly I would like to thanks my friends who gave me unending support and helped me in numerous ways from the stage when the idea of the thesis was conceived. I am very thankful to all of them for making my work complete successfully under their guidance.

Sarbjot Kaur
11109850

DECLARATION

I hereby declare that the dissertation entitled, **“Optimization of Regression Testing Technique for Web Application”** submitted for the M.Tech Degree is entirely my original work and all ideas and references have been duly acknowledged. It does not contain any work for the award of any other degree or diploma.

Date:

Sarbjot kaur

11109850

CERTIFICATE

This is to certify that Sarbjot Kaur has completed M.Tech dissertation proposal titled **"Optimization of Regression Testing Technique for Web Application"** under my guidance and supervision. To the best of my knowledge, the present work is the result of his original investigation and study. No part of the dissertation proposal has ever been submitted for any other degree or diploma.

The dissertation proposal is fit for submission and the partial fulfillment of the conditions for the award of M. Tech Computer Science & Engineering.

Date:

Name: Mr.Makul Mahajan

Assistant Professor

UID: 14575

TABLE OF CONTENTS

Page No.

CHAPTER 1: INTRODUCTION……………………………………………………………1

1.1 Software engineering……………………………..2

1.2 Software Testing...3

1.2.1 Verification...3

1.2.2 Validation..4

1.3 Software Testing Types...4

1.3.1 Black Box Testing...4

1.3.2 White Box Testing..4

1.4 Web Testing..5

1.5 Website Quality..5

1.6 Techniques of Web Testing..6

1.6.1 Functionality Testing……………………………….................................6

1.6.2 Usability Testing…………………………………………..........................6

1.6.3 Interface Tesing…………………………………………….......................7

1.6.4 Compatibility Testing……………………………………….......................7

1.6.5 Performance Testing..7

1.6.6 Security Testing...7

1.7 Regression Testing...7-9

1.7.1 Types of Regression Testing...10

1.8 Regression Testing Technique..10

1.8.1 Retest All...10

1.8.2 Regression Test Selection..10

1.8.3 Test Cases Prioritization..11-13

1.9 Method of Regression Testing..14-15

CHAPTER 2: Review of Literature...16-23

CHAPTER 3: Present Work..24

3.1 problem formulation..24

3.2 objectives...24-25

3.3 Methodology...25-26

3.4 Genetic Algorithm Steps...26-27

CHAPTER 4: Result and Discussion...28-40

CHAPTER 5: Conclusion and Future scope..41-42

CHAPTER 6: List of References...43-44

List of Figures

Figure No. **Page No.**

Figure1.1: Regression Testing..9

Figure3.3: Flow Chart of Our Methodology...26

Figure4.1: GUI start up...28

Figure4.2: Pop up box for choosing the file..29

Figure4.3: Output when both file are chosen..29

Figure4.4: Output of pre-process and extract variable...30.

Figure4.5: Output of compare button..31

Figure4.6: Output of extract constraints...31

Figure4.7: Output of compute reusability...32

Figure4.8: Output of generate path button...33

Figure4.9: Output of genetic algorithm...34

Figure4.10: Total number of path...35

Figure4.11: Total number of inputs..36

Figure4.12: Reusable input...36

Figure4.13: Reusability..37

Figure4.14: Reusability of first file...37

Figure4.15: Reusability of second file...38

Figure4.16: Reusability of third file..38

Figure4.17: Flow chart of previous work..39

Figure4.18: Flow chart of proposed work..40

List of table

Table 4.11 Reusability Table..34

CHAPTER 1

INTRODUCTION

1.1 Software Engineering

Programming architects apply the standards of programming designing to the outline, improvement, support, testing, and assessment of the product and frameworks that make PCs or anything containing programming work. Software designing is the study and a use of building to the outline, improvement, and support of software. Run of the mill formal meanings of programming designing are the use of a methodical, trained, quantifiable way to the improvement, operation, and support of software a designing teach that is concerned with all parts of programming production also the foundation and utilization of sound designing standards to monetarily acquire programming that is dependable and lives up to expectations productively on genuine machines. Because of innovative development and aggressiveness in business, programming continues evolving. The changed programming must be tried completely with the purpose that the changed bit of code does not influence different parts of the code. Regression testing is a suitable testing method to be connected at this phase of the product testing. The primary destination of relapse testing is retesting the changed parts and checks the influenced parts of the product. It is a critical and exceptionally testing assignment for the product analyzers to test the whole programming inside the restricted time and assets, To lead relapse testing more successfully the experiment prioritization systems are utilized. The experiment prioritization methods enhance the expense adequacy of relapse testing by requesting experiments such that those that are more essential are run prior in the testing methodology. In the writing a few exploration meets expectations have been completed to organize the experiments more successfully. The current work is in light of certain scope criteria, for example, code, flaw, prerequisite, and so on and an experiment with most extreme scope is chosen by contrasted and all other experiments.

Programming testing assumes an imperative part in guaranteeing the nature of programming frameworks. Notwithstanding, it is evaluated that product testing expends more than 50% of the expense in programming improvement and upkeep. In this manner, numerous specialists concentrate on the best way to mechanize programming testing furthermore consequently enhance the effectiveness of programming testing. Experiment prioritization. By reusing the experiments of its past rendition (some time recently change). To encourage relapse testing, experiment prioritization plans the execution request of experiments so as to expand some target (e.g., un-covering blames early). Given a past system P and its adjusted variant P', the methodology of regression testing incorporates building a test suite for P' in light of the current test suite

1.2 Software Testing

Programming testing is the methodology by which assessment of software is done to identify that whether their falsehoods any contrasts between the given information and expected yield or not. Through this procedure evaluation of the quality of a product thing is finished . Currently the question is "when it ought to be done?" Most of the developers and researchers say that programming testing ought to be done due to improvement process. The methodology programming testing is performed to confirm that whether programming item is assemble as per the client necessities or not. currently talking regarding its objectives, the most objective of programming testing is to search out errors in the system/software being developed or changed by applying different types of testing like performance testing, regression testing, smoke testing, stress testing. By code testing, the standard parameters like effectiveness of programming are often improved. It also can be said that software testing is each verification and validation. currently what's meant by these2 terms?

1.2.1 Verification

Verification suggest that "are things right or not?" There forever stay some conditions that are obligatory on the software at the beginning of the improvement stage.Confirmation is the methodology to verify the item fulfils each one of those conditions .In different words, to verify that the item carries on the way we need it to is called verification.

1.2.2 Validation

Validation means “are we have a tendency to doing right things?”. Through validation method, we make sure that at the end of development phase, product meets the specified requirements. In alternative words, validation is to make sure the product/system is built as per needs of the customer or not.

1.3 Software Testing Types

To make the things clear some of the types of software testing are described here. In this proposal, main focus is on regression testing .

1.3.1 Black Box Testing

Black box testing that is additionally known as functional testing is a testing technique within which focus isn’t on the interior mechanism of the system. Its fundamental center stay on the yield created against any input give to the system.

1.3.2 White Box Testing

Rather than the functional box testing, in the event of structural testing stay on the code i.e. it is a testing procedure that consider the inside system of a framework. It is additionally structural testing or glass box testing.

“Black box testing is commonly used for validation and white box testing is commonly used for verification.

There are some more approaches of testing which are:

a) Unit Testing
b) Integration Testing
c) Functional Testing
d) System Testing
e) Stress Testing
f) Performance Testing
g) Usability Testing
h) Acceptance Testing
i) Regression Testing
j) Beta Testing   :

1.4 Web Testing

Web testing is that the name given to package testing that focuses on internet applications. Complete testing of a web-based system before going live will facilitate address problems before the system is disclosed to the general public. problems like the safety of the online application, the essential practicality of the location, its accessibility to incapacitated users and absolutely ready users, stills readiness for expected traffic and variety of users and therefore the ability to survive a vast spike in user traffic, each of that square measure associated with testing . Making an internet web site doesn't finish with swing all the media and software system along. Actually, information processing system work ne'er ends. Once all the planning is finished, you have got to check the positioning 1st before causation it to the planet Wide net for the planet to check. There's web site management software system that may try this for you. These software system will facilitate reconnect graphics which will are accidentally moved , modification the name of a file and re-link it then several different things. Apart from the positioning management software system, you furthermore might need to the standard of your web site. Your web site needs to be tested, fixed, and retested and absolutely documented [9].

1.5 Website Quality

There are thousands of websites launched once a year and zilch is worse than a poorly operative website. Website testing is most vital to e-commerce sites since they have applications running on the website that will have an effect on their sales or operations.

Thus however websites should be tested? What are the measures of quality? Here are a number of measures of quality according to Miller:

a) Timeliness n WebPages got to be upgraded continuously. Once it absolutely was last upgraded? How consistent is it to now day be news or information?
b) Structural Quality ñ all the components of the website got to be operating well. All the links (inside or outside) working? Are all the images loading?
c) Content ñ This doesn't merely concern spell checking, proofreading but also the regularity of the contents with either all of the opposite pages or with the request of the user, particularly with dynamic sites. Will the content of important pages match what is

purported to be there? Do key phrases exist frequently in highly- changeable pages? Do important pages maintain quality content from version to version?

d) Correctness and reliability n are the pages consistent with what the user requested? However consistent is that the webpage with yesterday's webpage?

e) Response Time and Latency ñ this is often essential with e-commerce sites. The latency of the server ought to be quick once clicking SUBMITS. Will the Website server reply to a browser request inside bound performance parameters? Are there parts of a site that are so sluggish the user discontinues working?

f) Performance ñ this involves performance by load or usage. Is that the webpage loading in but eight seconds? Will your system handle ten,000 transactions per minute?

g) Quality of the website is very vital for the user. An internet site with too several broken

   a. Links, defective picture, could price plenty for an e-commerce website. Users can quickly

   b. Leave for a dissimilar web site if the website is simply too advance and of inferiority [10].

1.6 Techniques of Web Testing:

Here are a few of the basic testing techniques for web applications:

1.6.1 Functionality Testing

The main goal of functional testing is to create positive that each one the functions among an online application area unit operating swimmingly with none technical glitches. in a very net application, useful testing may cowl various things like whether or not all the links area unit operating properly or not, testing forms altogether the pages, testing cookies, sustentative HTML or CSS, testing information for the safety so on. One ought to additionally make sure that take a look at cases cover all the boundary conditions that require to be tested.

1.6.2 Usability Testing

When it involve to create the application easy and effective, its user interface ought fulfill with the standards. It is vital that you simply follow all the world conventions and net standards whereas developing an online application. Usability testing is appropriate for the applications that are proposed to streamline the manual process. However, one should also keep in mind certain essential aspects like correct navigation, site map, avoid via over-crowded content and additional whereas approaching for the usability testing.

1.6.3. Interface Testing

one of the for most vital interfaces among internet application are web server and application server interface and database server interface. Interface testing can make ensure that all the individual elements among an application are connected properly. One ought to check whether or not the interaction between these servers are executed properly or not with the help of this testing technique [9].

1.6.4. Compatibility Testing

Compatibility of your net application is one in every of the formost crucial belongings you ought to consider whereas testing the appliance. Compatibility testing can check your website or net application for browser compatibility, operating system compatibility, mobile browsing and printing choices.

1.6.5. Performance Testing

Performance testing can assist you confirm the performance of your internet application under various scenarios. Performance testing sometimes involves stress testing, scalability testing and load testing. In this testing method, website istypically tested for its practicality on totally different operating system, hardware platforms and additional.

1.6.6. Security Testing

This testing methodology is one in every of the formost necessary ones for your net application as if information leaks or modifications are unit tolerable or not. It alwayes involves varied things like testing the CAPTCHA for automates scripts logins, testing SSL for security measures, whether or not it’s feasible to access net directories or files directly or not and then on.

1.7 Regression Testing

Regression testing means that re-testing an application when its code has been changed to verify that it still functions properly. Regression testing consists of re-running existing test cases and checking that code changes didn’t break any earlier operating functions, unconsciously introduce errors or cause earlier mounted problems to re-emerge [2]. These test cases ought to be run as usually as attainable with an automatic regression testing tool, so code modifications that injury however the application work is quickly known and set. Regression testing starts as shortly there's something to test in any respect. .The regression test suite grows because the application moves ahead and check engineers add check cases check test new or rewritten code. presently the suite could touch thousands of check cases that face various application functions. Obviously, by this time, automation of regression check becomes vital as a result of it's humanly not possible to quickly and dependably repeat all of the test cases and analyze their results. This is often wherever Test Complete is incredibly useful and provides you with most come back on your machine-driven testing in Regression testing is a kind of code testing that seeks to uncover new code bugs, or regressions, in existing practical and non-functional areas of a system once changes like enhancements, patches or configuration changes, are created to them. The intent of regression testing is to confirm that changes like those mentioned on top of haven't introduced new faults.[1] One amongst the most reasons for regression testing is to see whether or not a modification in one apart of the software affects alternative components of the software package. Common way of regression testing embrace rerunning earlier completed tests and checking whether or not program behavior has modified and whether before fixed faults have re-emerged. Regression testing are often performed to check a system expeditiously by consistently choosing the suitable minimum set of tests required to adequately cowl up a selected modification..

Contrast with testing, that aims to verify whether or not, when introducing or change a given code application, the modification has had the supposed impact. Regression testing may be a variety of code testing that decide to sight new faults or bugs in software when modification have been made in existing practical and non-functional areas of package. Regression testing is is additionally a verification method that determines that previous character and functionality of software remains when a modification is created within the code.. Its main goal is to confirm that no new error are introduced into antecedent tested code or package when changes are created within the code One of the most reasons to introduce regression testing is to work out whether or not a modification in one part of the computer code affects alternative components of the computer code. Let P be a program [3], let P′ be a changed version of P, and let „T" be a check suite for P. Regression testing consists of reusing T" on P′, and wherever wherever the new check code or practically accessorial to or modified in manufacturing P′. Regression testing could be a necessary feature of the intense programming software development technique. Even with in the company world, regression testing has been performed by a computer code quality assurance team.

Regression testing is employed not just for testing the correctness of a program, however usually additionally to trace the standard of its output [4]. As an example within the style , of a compiler, regression testing may trace the dimensions of code, and therefore the time it takes to compile and execute the complete take a look at suite cases. Regression testing may be a meaningless method happens at an optimum price and in minimum time. The aim behind regression testing is extremely easy and straight. It will increases the productivity and potency of software package and quality assurance applications..

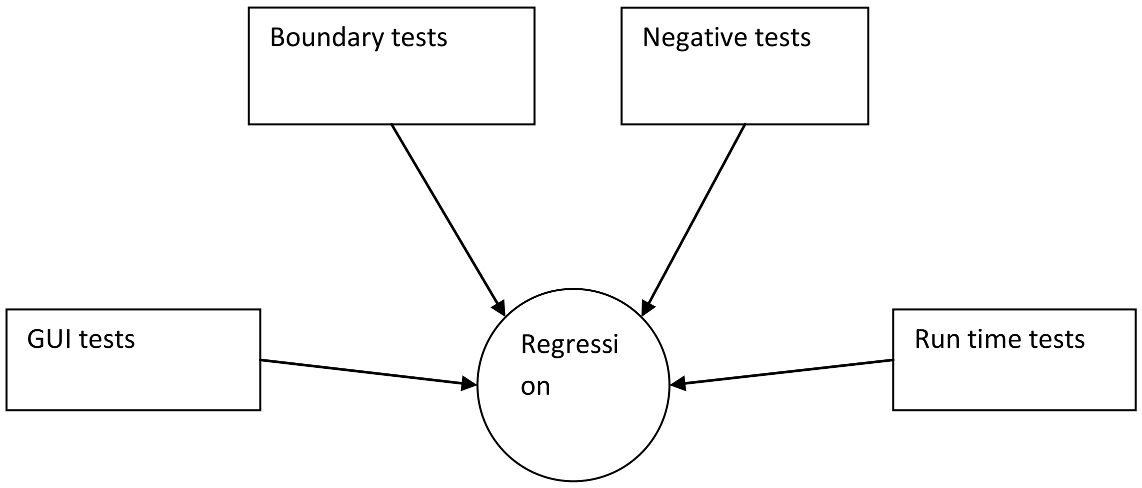


Fig 1.1: Regression Testing

The expectation of regression testing is to ensure that those progressions, for instance, those specific on top of haven't given new faults. One of the principle functions behind relapse testing is to work out if a amendment in one piece of the manufactured goods influences dissimilar parts of the software.

Basic techniques for regression testing incorporate rerunning already finished tests and checking whether or not program conduct has modified and whether or not beforehand altered shortcomings have re-developed. Regression testing are often performed to check a framework effectively by deliberately selecting the fitting least set of tests expected to sufficiently cover up a selected amendment .

1.7.1 Types of Regression Testing:

- **Final Regression Tests: -** A "final regression testing" is performed to validate the build that hasn't modified for a some amount of time. This build is deployed or shipped to customers.
- **Regression Tests: -** A standard regression testing is performed to verify if the build has NOT broken the other elements of the application by the recent code changes for defect fixing or for improvement [7]

1.8 REGRESSION TESTING TECHNIQUES

Regression testing has been categorized to different testing techniques so that maintenance testing process can be performed effectively and easily with minimum time and cost requirements. Regression testing techniques are given below:

1.8.1 Retest All

Retest all methodology or technique is one among the normal techniques for conducting regression testing. Because the name indicates retest all is that the techniques during which all the tests are arrunned and executed once more with in the existing test suite functional and non-functional areas. So the retest all technique [5] is extremely big -ticket and time consuming. As regression test suites are expensive to execute fully because it need longer time and budget alternative techniques like Regression Test Selection and Test Case Prioritization are required to be introduced to create regression testing price and time effective.

.1.8.2 Regression Test Selection

Due topic nature of “retest all” technique, Regression Test Selection (RTS) is performed. during this technique rather than rerunning and perform the entire test suite. we select a part of test suite to rerun provided the cost of selecting a part of test suite is less than the cost of running the all tests that RTS allows us to omit. RTS divides the existing test suite into following test cases:

- Reusable test cases
- Retestable test cases
- Obsolete test cases

In addition to this classification RTS may create new test cases that test the program for areas which are not covered by the existing test cases. RTS techniques are broadly classified into three categories [6]:

a) Coverage techniques: They take the test coverage criteria under consideration. They search coverable program elements that are changed and choose test cases that work on these elements.
b) Minimization techniques: They're same as coverage techniques except that they choose minimum set of test cases
c) Safe techniques: They do not specialize in criteria of coverage; in distinction they choose all those take a look at test cases that turn out completely different output with a changed program as compared to its original version. In regression test selection, test cases are selected because their execution is relevant to the modified between the earlier and the present version of the software system.
d) Inclusiveness: It is the calculate of extent up to that a method chooses the test cases which is able to cause the modified program to supply totally different output than the original program, leading to exposure of faults during modifications.
e) Precision: It is the determine the ability of technique to prevent selecting test cases that may not create the modified program to supply totally different output than the unique program.
f) Efficiency: It measures the utility (computational cost) of a method.
g) Generality: It's the measure of ability of a method to handle complicated modifications, realistic language constructs and realistic testing applications.

1.8.3 Test Case Prioritization

a) Test Cases

1. A **test suit**, in computer code engineering, could be a set of conditions beneath that a tester will confirm whether or not an application, software system or one in every of its features is working as it was initially established for it to try do.

b) Types of Test Cases

- **Formal test cases**

In order to completely check that all the needs of an application are met, there should be a minimum of must t two test cases for every requirement: one positive test and one negative test. If a demand has sub-requirements, every sub-requirement should have minimum of 2 test cases. Keeping track of the link between the need and therefore the test is usually done using a traceability matrix. Written test cases ought to embody an outline of the practicality to be tested, and the preparation needed to make sure that the test can be conducted. A formal written test-case is characterized by a known input and by an expected output, which is worked out before the test is executed. The known input ought to check pre condition and therefore the expected output ought to test a post condition.

- . **Informal test cases**

For applications or systems with no formal needs, test cases often written depend on accepted simple operation of programs of a same class. In some area of testing, test cases don't seem to be written in the slightest degree however the activities and results are measure reported when the tests are run . In situation testing, theoretical stories are use to aid the tester think during a difficult problem or system. These scenario are sometime not written down in any detail. They'll be straight forward as a diagram for a testing surroundings or they might be a outline written in prose. The best situation test could be a story that is motivating, credible, complex, and easy to calculate. Sometime completely different from test cases in that test cases are single steps whereas situation cover a variety of steps of the key.

c) Test Case Prioritization

The test case prioritization problem is a research hotspot in the field of software testing. It is defined as follows:

- **Coverage Prioritization:** It counts the overall variety of statements covered by each test case, and then kind the action in line with the quantity. However, this strategy might create statement coverage of a test case be a subset of an extra test coverage, and eventually, there are still some statements that haven't been enclosed by any test cases. Therefore, extra strategy is appeared. It initial picks the test case with the best coverage, and then, in turns adds those test cases that face the formost however uncovered elements.

- **History Prioritization:** This technique prioritizes test cases based on historical execution knowledge. This method helps to scale back spending. With in the long standing time, it will enhance the effectiveness of regression testing. For a test case, using past data of each prioritization, increase or decrease its probability with in the the current test session. A likelihood price is given to the the past and the value of the present session is calculated supported the quantity of fault detection (or coverage). Finally, the calculated value of a test case is equal to the sum of the present number multiplied by a probability and the historical price increased by another likelihood.

- **Time Spending Prioritization:** This method is that count execution time of each test case and then sorts the test suite according to the ascending order of the calculated value.

Test Case Prioritization (TCP) technique of regression testing prioritize the test cases therefore on increase a test suits rate of error detection that's however rapidly a test suite find faults with in the changed program to extend reliability. This is of 2types:
a) General prioritization that makes to pick out an order of the test case that may be effective on average sresultant versions of software.
b) Version Specific prioritization that worries with explicit version of the software.
Test Case Prioritization are often classified any as given below:

a) Comparator techniques: These accommodate random ordering and best ordering.
b) Statement level techniques: These techniques also are called Fine Granularity. They accommodates of total statement coverage prioritization, extra statement coverage prioritization, total fault-exposing-potential (FEP) prioritization further FEP prioritization.
c) Function level techniques: These techniques also are called as Coarse Granularity. They accommodates whole operate coverage prioritization, extra function coverage, total FEP

prioritization, more FEP prioritization and total fault index (FI) prioritization, additional Fault Index (FI) prioritization, whole FI with FEP coverage prioritization and other FI with FEP coverage prioritization.

- **Hybrid Approach**

Hybrid Approach of regression testing because the name suggests is that the hybrid of each Regression Test Selection and Test Case Prioritization. This approach consist features and properties of each test case selection and test case prioritization techniques. It is an well-organized approach. There are range of researchers performing on this approach and that they have planned several algorithms for it. For instance,

a) Test Selection Algorithm: planned by Agawam et al. Implementation of algorithm [11]:

- Input
- Test Selection algorithm: Adjust module and decrease module
- Output.

b) Hybrid technique planned by Wong et al which mixes decrease, changes and prioritization based mostly choice exploitation selection using test history.

1.9 Method of Regression Testing

a) Run the entire existing test: The most wide used regression strategy is to run all of the previous tests. If there's a test that has been written, and it absolutely was run before, it goes into the regression suite. However this strategy has a few downsides. If all those tests are manual – when running identical tests 4-5 times, the testers are attending to be bored and exhausted. If the regression suite includes all of the presented tests, it's attending to be monumental. Even once a regression suite contains all of the tests, and they all pass, this doesn't mean there aren't defects. Testing an application one hundreds percent is perhaps not possible.

b) Run test that are at high risk: Another methodology used for regression suites is to implements tests that area unit high threat. The tests you have to run ought to be organized by the business users. Tests that are the mainly vital to the business clients area unit are those processes the business users do constantly—and the usefulness is discriminating to them. Keep in mind that as the application changes, the key business techniques might likewise change. It is critical to determine a period confine in your test arrangement for the high-hazard tests. Depending on what else needs to be incorporated in the regression suite, 30-40 percent of your total regression time is often assigned for high risk tests.

c) High Defect Feature: The third technique is high-surrender highlights. This tests common range of the application that are high desert regions or zones that are extremely advance . May be it's simply the highlight that is complex– it may incorporate complex estimations or joining with one or more different applications. This additionally incorporates functionalities that have had several defects within the past.

d) Exploratory Testing: The final technique is named exploratory testing – this is often NOT random testing. True random testing are often done by anyone – with no information of the application or testing – it's all random. Exploratory testing is regarding doing test suit style and execution at a similar time. As you produce and execute these tests, you find issues and features inside the application. These revelations drive what is tried next. Verify there are sufficient notes for you and the designer to replicate any issues. Exploratory testing exploits the analyzer's experience and accepting of application testing.

e) Automation Testing: If you've got variety of regression tests, automation will be brilliant to reduce the quantity of tests to run yourself. Automation tools, like HP's Unified Functional Testing, can run the tests fastly. Therefore, this reduces the quantity of your time required to implement the similar test throughout the regression testing section. A manual test will take five minutes to implements; however that very same test could take less than a minute to implements through Unified Functional Testing. This could assist you to induce a additional comprehensive regression test suite. However, automation isn't a fast fix. Automation scripts take time to develop. You 've got to permit for time between regression runs to develop and obtain the tests operating. If your surroundings is dynamic, automation scripts can ought to updated consequently [8].

CHAPTER 2

REVIEW OF LITERATURE

Reetika Nagar *et al* says Maintenance of software is a very crucial and important task but it is a very expensive process. Therefore, Software Maintenance Testing is important throughout package testing section. Error and defect found throughout testing method should undergo a re-check the method so that flaws, that is, defects and errors can be eliminated simply. By doing thus, check cases are fully required to evolve and modify in line with the ever-changing needs. In this paper, a brief introduction of several maintenance testing's such as confirmation testing and regression testing; and introduction of maintenance testing techniques has been given. These techniques are to be used in testing during software development to make testing process effective.[1]

Thillaikarasi Muthusamy *et al* says Regression Testing is that the method of running the test cases which have passed on the previous build or release of the application under test in order to validate that the original features and functions are still working as they were previously. It's impossible and in-sufficient resources to re-execute each action for a program if modification occurs. This downside of regression testing often solved by prioritizing test cases. A regression test case prioritization technique involves re-ordering the execution of test suite to extend the speed of fault detection in past stages of testing method. In which paper, test case prioritization algorithm is planned to find the severe faults and progress the speed of fault detection. This is planned test case prioritization algorithm prioritizes the test cases supported four team of practical weight issue such as: customer allotted priority, developer observed code execution complexness, changes in necessities, fault impact, fullness and traceability. [2]

Hossain, M *et al* says Companies that provide web applications need to perform frequent regression testing as result of corporations usually encounter numerous security attacks and common feature inform demands from users. Typically, such applications require regression testing processes that require minimal test effort because they have already been deployed and used in the field. In our previous work, we presented an efficient regression testing approach that enable us to focal point to target the areas of code that are modified and to regression test them to deal with this drawback. Whereas our experiment results showed that this method is economical in saving the cost of regression testing by dropping the amount of test paths necessary for the changed program, we also learned that resolving input constraints requires a lot of effort. In this paper, to accommodate further savings with regression testing, we have a tendency to present a way that finds the reusable constraint values for regression test cases.[3]

Wenhong Liu *et al* says The adequacy of the regression testing and the minimization of the test case suite are the important parts of the research in the field of software testing. In this paper, for the characteristics of regression testing, a regression test case techniques supported the analysis of the connection is planned. By analyzing the relationship between the testing requirements, testing requirements and test cases, test cases and software faults, software faults and software changes, with the priority of the testing requirements and test cases, it achieves the function of screening the regression test case suite. At the same time, it provides a wizard for the test case design of the new testing requirements. This method improves the efficiency and adequacy of the software regression testing effectively.[4]

McMaster, S *et al* says Making machine controlled tests that exercise an internet application through a browser may be a difficult and long method. During this paper, we describe a new tool presently underneath open-source progress, internet Testing adventurer, that uses runtime state from the web application as feedback to search for defects in real-time and automatically create increasingly longer test cases with oracles for later execution while providing testers with the flexibility needed to deal with numerous web testing challenges.[5]

Leotta, M *et al* says There are several approaches for automated functional web testing and the choice among them depends on a number of factors, including the tools used for web testing and the costs associated with their adoption. In this paper, we present an empirical cost/benefit analysis of two different categories of automated functional web testing approaches: (1) capture-replay web testing .On a set of six web applications, we evaluated the costs of applying these testing approaches both when developing the initial test suites from scratch and when the test suites are maintained, upon the release of a new software version. Results indicate that, on top of the one hand, the development of the test cases is more expensive in terms of time required when the programmable web testing approach is adopted, but on the other hand, test suite maintenance is less expensive when this approach is used .We found that, in the majority of the cases, after a small number of releases ,the cumulative cost of programmable web testing becomes lower than the cost involved with capture-replay web testing and the cost saving gets amplified over the successive releases.[6]

Lei Xu *et al* says Net application testing is worries with various and sophisticated testing objects, strategies and processes. Thus a testing framework fitting for the properties of Web application is needed to guide and organize all the testing tasks. Supported the analysis for net application characters and fixed software testing method, the process for net application testing is modelled, that describes a series of testing flows like the testing demand analysis, check cases generation and choice, testing execution, and testing output analysis and measuring.[7]

N. Prakash1 *et al* says Cost and time effective reliable test case prioritization technique is the need for present software industries. The 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 alleviate the above problem a new methodology using modular based prioritization test cases is proposed for testing of regression. 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 planned process is assess using 3 standard applications called Hospital Management System, University Students Monitoring System and Industrial Process Operation System. The empirical studies illustrate that the planned algorithm is significantly performed fine. The superiority of the proposed method is also highlighted**.[8]**

Xiaolin Wang *et al* says Test case prioritization innovation is to sort the experiments before the product testing intended to enhance test productivity. This paper displays an element experiment prioritization procedure taking into account multi-objective. It incorporates a few customary single-target innovations with the goal that makes it more adaptable. This innovation, from five measurements, figures prioritization estimations of experiments independently. At that point a weighted entirety is made to the qualities and it sorts the experiments as indicated by the qualities. The outcomes come back to the capacity keeping in mind the end goal to rapidly alter the kind of experiments. This innovation not just takes care of the levels of popularity of relapse testing, additionally guarantees the high effectiveness of the test outcomes.[9]

Chu-Ti Lin *et al* says Test case prioritization procedures plan the experiments in a request in light of some particular plane then tests with best blame discovery ability are running at an starting position within the relapse test case. Numerous earlier experiment prioritization methodologies are code-based, by which the testing of every product form is measured as an autonomous procedure. Really, the test after-effects of the former programming renditions will be valuable for booking the experiments of the future programming forms. A few scientists have planned past-based ways to deal with issue, they established that the quickly going before test outcome gives the similar location worth to organizing the experiments of the progressive programming form over the whole lifetime of the product advancement process. Consequently, this paper depicts continuous exploration that study that whether the reference estimation of the promptly going before test outcomes is variant mindful and propose an experiment prioritization methodology in light of our perceptions. The trial results show that, in examination to existing methodologies, the exhibited one can calendar experiments all the more successfully.[10]

Bo Jiang *et al* says Test case prioritization allocates the running needs of the experiments in a given test suite with the point of attaining to specific objectives. Numerous existing experiment prioritization methods however expect the undeniable accessibility of code scope information, issue history, or test detail, which are sometimes decently kept up in numerous programming advancement ventures. This paper proposes a novel group of LBS systems. They make versatile tree-based randomized investigations with a versatile randomized hopeful test set method to differentiate the investigations among the extensions of the investigation trees built with the test inputs in the test cases. They dispose of the presumption on the chronicled connection of code scope between system adaptations. Our methods can be connected to projects with or with no past variants, and thus are more general than numerous existing experiment prioritization procedures. The exact study on four prominent UNIX utility benchmarks demonstrates that, as far as APFD, our LBS procedures can be as compelling as a portion of the best code scope based avaricious prioritization systems ever proposed. We likewise demonstrate that they are altogether more productive and adaptable than the last procedures.[11]

Arafeen, M.J. *et al* says When programming frameworks develop, diverse sums and sorts of code alterations can be included in distinctive adaptations. These variables can influence the expenses and profits of relapse testing systems in diverse ways, and consequently, there might be no particular relapse checking method that is the much savvy strategy to use on each form. To date, numerous relapse testing procedures have been proposed, however no exploration has been carried out on the issue of helping professionals efficiently pick fitting strategies on new forms as frameworks advance. To address this issue, we propose versatile relapse testing (ART) methods that endeavour to recognize the relapse testing strategies that will be the most practical for every relapse testing session considering association's circumstances and testing environment. To evaluate our methodology, we led a trial concentrating on experiment prioritization strategies. Our outcomes demonstrate that prioritization methods chose by our methodology can be more practical than those utilized by the control approaches.[12]

Harrold, M.J *et al* says Regression testing is an essential movement that can represent a vast extent of the expense of programming upkeep. One way to decreasing the expense of relapse testing is to utilize a particular relapse testing system that: picks a division of a test case that was utilized to check the product before the alterations; then uses this subset to test the changed programming. Specific relapse testing procedures diminish the expense of relapse testing if the expense of selecting the subset from the test suite together with the expense of running the chose subset of experiments is not exactly the expense of rerunning the whole test suite. Rosenblum and Weyuker (1997) proposed scope based indicators for utilization in anticipating the adequacy of relapse test choice techniques. Utilizing the relapse testing expense model of Leung and White (1989; 1990), Rosenblum and Weyuker showed the appropriateness of these indicators by performing a contextual investigation including 31 variants of the Korn Shell. To further research the appropriateness of the Rosenblum-Weyuker (RW) indicator, extra experimental studies have been performed. The RW indicator was connected to various subjects, utilizing two diverse particular relapse testing instruments, Deja vu and Test Tube. These studies help two conclusions. Initially, they demonstrate that there is some variability in the accomplishment with which the indicators work and second, they propose that these outcomes can be enhanced by fusing data about the dispersion of changes. It is demonstrated how the RW expectation model can be enhanced to give such a bookkeeping**.[13]**

Liang You *et al* says According to this paper author found that after the developer fixes the bugs and improves the usefulness of the product venture, relapse testing reruns the relapse testing suite to guarantee that the new form programming undertakings can run easily and effectively. Since the relapse testing is the most extravagant period of the product testing, regression testing lessening takes out the excess experiments in the relapse testing suite and spares the expense of the relapse testing. This paper formally characterizes the time-mindful relapse testing diminishment issue. It additionally proposes a novel hereditary calculation for the time-mindful relapse testing lessening issue. It characterizes the representation and wellness capacity of the hereditary calculation; it likewise portrays the guardian determination, hybrid and change administrator of the hereditary calculation. The novel calculation not just uproots excess experiments in the relapse testing suite additionally minimizes the aggregate running time of the staying experiments. At long last, the paper assesses the hereditary calculation utilizing eight case programs. The exploratory result shows the productivity of the proposed hereditary calculation for the time-mindful relapse testing lessening issue.[14]

Xiao Qu *et al* says According to this paper configurable frameworks that let clients modify framework practices are getting to be progressively predominant. Testing a configurable framework with every single conceivable arrangement is extremely extravagant and regularly unfeasible. For a solitary variant of a configurable framework, inspecting methodologies exist that select a subset of arrangements from the full setup space for testing. In any case, when a configurable framework changes and develops, existing methodologies for relapse testing select all setups that are utilized to test the old forms for testing the new form. As exhibited in our examinations, this retest-all methodology for relapse testing configurable frameworks ends up being profoundly excess. To address this repetition, creator proposes a setup choice methodology for relapse testing. Formally, given two forms of a configurable framework, S (old) and S' (new), and given an arrangement of arrangements CS for testing's, our methodology chooses a subset CS' of CS for relapse testing S'. as indicated by their study comes about on two open source frameworks and a vast mechanical framework demonstrate that, contrasted with the retest-all approach, our methodology disposes of 15% to 60% of designs as excess. Their methodology likewise spares 20% to 55% of the relapse testing time, while holding the same shortcoming discovery capacity and code scope of the retest-all methodology.[15]

Engstrom E. *et al* says History based regression testing was proposed as a premise for robotizing relapse test determination, with the end goal of enhancing straightforwardness and test effectiveness, at the capacity test level in a substantial scale programming advancement association. The study goes for researching the flow manual relapse testing process and additionally receiving, actualizing and assessing the impact of the proposed technique. Strategy: A contextual investigation was dispatched including: distinguishing proof of critical components for prioritization and choice of experiments, execution of the technique, and a quantitative and subjective assessment. As indicated by creator comes about 10 separate components, of which two are history-based, are recognized as critical for determination. The greater part of the data required is accessible in the test administration and lapse reporting frameworks while some is inserted simultaneously. Straightforwardness is expanded through a semi-computerized system. Their quantitative assessment demonstrates a plausibility to enhance effectiveness; while the subjective assessment underpins the general standards of history-based testing however recommends changes in usage subtle elements.[16]

Shiming Sun *et al* says According to this paper relapse testing is a critical procedure amid programming improvement. With a specific end goal to lessen expenses of relapse testing, research on advancement of plan of relapse testing have been done in this paper. With the end goal of diminishing the quantity of experiments and identifying flaws of projects early, this paper proposed to consolidate experiment choice with experiment prioritization. Relapse testing procedure has been composed and streamlining of testing plan has been executed. The standard of experiment determination is alter effect of projects, discovering projects which are affected by system adjustment as indicated by adjust data of projects and conditions between projects. Experiments would be chosen amid experiment choice. The rule of experiment prioritization is scope capacity and investigating abilities of experiment. Experiments which have been chosen amid experiment choice would be requesting in experiment prioritization. At long last, the viability of the new system is talked about.[17]

Kayes M.I. *et al* says According to this paper experiment prioritization procedures include planning experiments for relapse testing in a request that builds their viability at gathering a number of execution objective. This is wasteful to re running all the experiments in relapse checking after the product adjustments. Utilizing data got from past experiment execution, prioritization methods arrange the experiments for relapse testing so that most gainful are executed first consequently permits an enhanced viability of testing. One execution objective, rate of reliance distinguished among issues, measures how rapidly reliance among shortcomings are recognized inside the relapse testing methodology. An enhanced rate of flaw reliance can give speedier criticism on programming and let designers begin troubleshooting on the extreme blames that cause different flaws to seem later. In which paper introduces the original metric for evaluating charge of issue reliance discovery and a calculation to organize experiments. Utilizing the new metric the viability of this prioritization is indicated contrasting it and non-organized experiment. Examination demonstrates that organized experiments are more viable in distinguishing reliance among issues.[18]

.

CHAPTER 3
PRESENT WORK

In this chapter, we are going to present the problem of our research work, its objectives, the methodology that we used for our proposed approach and the introduction of the tool which will be used to complete the task. In the 3.1 section we explain how we formulated our problem and what the approach we are going to use. In the 3.2 and 3.3 section the objectives and methodology of the work done. In the methodology the flow of our work with the help of flow chart is explained.

3.1PROBLEM FORMULATION

Web testing is used widely in the various fields of computers and business. The uses of various companies have been increasing rapidly. The changing in their formats and requirements changes day by day. 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.

3.2 OBJECTIVE

The reusability causes failure error due to rapid changes in the web applications. In these applications constraints value reuse must be optimized and validated.

The main objectives of research study are:

- To develop different test cases for web testing on the basis of priority and functionality.
- To implement constraint reusability using previous version of test suites.
- To validate different test cases using checksum approach.
- To optimize the test case sequence by using genetic algorithm.
- To increase the reusability of constraints and test cases.
- To validate purposed work comparison between purposed and previous work on the basis of parameters

3.3 METHODOLOGY

In this work various test cases have been used for the purpose of regression testing when some changes has been made in the previous developed system. These test cases has been developed using different priorities and constraint values. Due to vast changes occurred in various sites of different online marketing services providers, regression testing has to be performed again and again. The test cases generate again and again takes time and cost effort. To reduce this effort reusability of various test case constraints has been used to perform regression testing of changed web application. Reusability of different test cases has been done by using different test cases prioritizations on the basis of these various constraints has been developed by on the basis of previous values and new constraints has to be used for checking of which changes has been made does not affect previous applications. These test suits have to be optimized on the base of different type of optimization approaches using different types of optimization approaches.

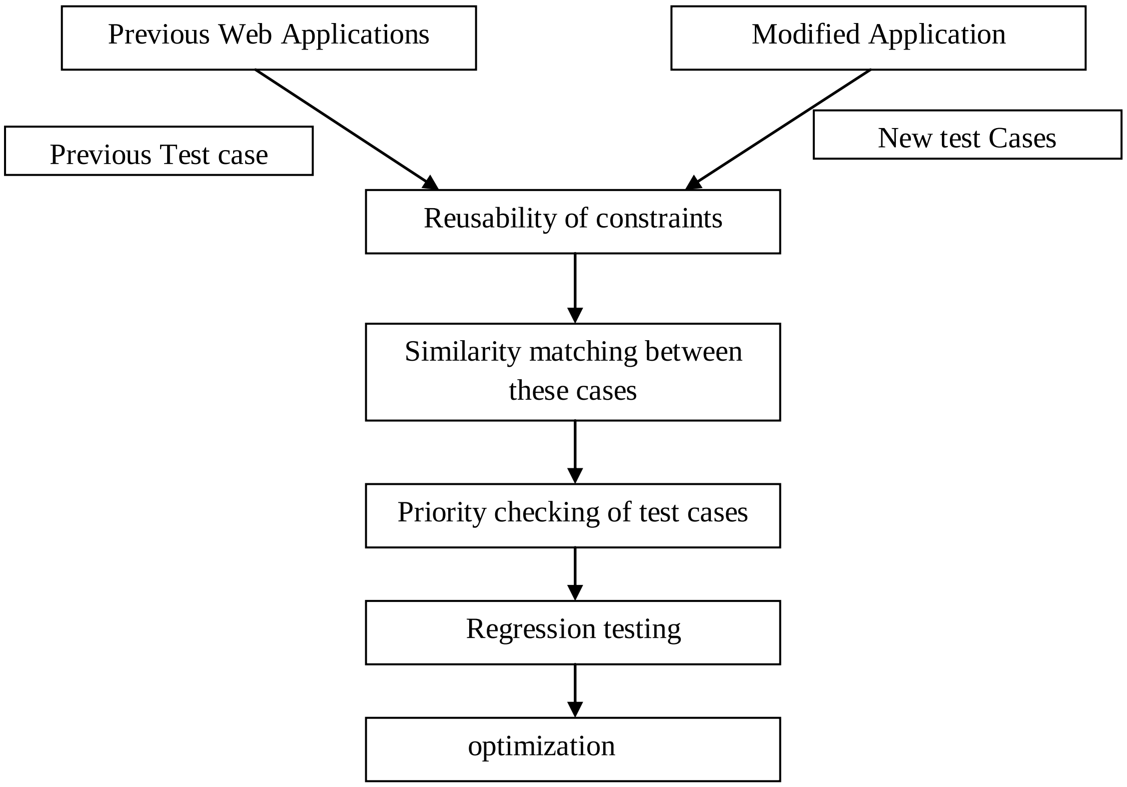

Previous Web Applications
Modified Application
Previous Test case
New test Cases
Reusability of constraints
Similarity matching between these cases
Priority checking of test cases
Regression testing
optimization

Figure 3.3 flow chart of methodology

3.4 Genetic Algorithmic Steps

Input – Test sequence coverage area, Number of test cases

Output – Best test case execution sequence

```
Begin
Initialize population
Select random candidate
Evaluate each candidate
int crossover probability
int mutation probability
for i=1; number of test case
generate random execution sequence
end
for i=1:n
```

```
chromosomes = generated sequence
for each chromosome
select two parents randomly
crossover operator
if (child) ≠ parent : add to generation
end if
end for loop
for each chromosome
select 1 test case from n sequence
child – mutation (chromosome)_
end for loop
evaluate fitness of new generation
for 1 to no. of child generation
CA = compute coverage area for each sequence of test cases
Sort (CA)
End
% select best sequence
Select the sequence having maximum (CA)
Best test case sequence = maximum convergence for sequence
End.
```

CHAPTER 4

RESULTS AND DISCUSSION

In this chapter we discussed the results of our research.

4.1 Tool Used

In our study we used eclipse tool to calculate the reusability. Eclipse is an integrated development environment (IDE). It contains a base space and an protrusible plug-in system for customizing the surroundings. It written mainly in Java, Eclipse may be wont to develop applications. By means that of varied plug-INS, Eclipse might also be wont to develop applications in different programming languages It may be wont used to create packages for the software Mathematic. The Eclipse package development kit (SDK), which incorporated the Java development tools, is supposed for Java developers. Users will expand its talents by putting plug-ins written for the Eclipse Platform, like development toolkits for different programming languages, and may write and contribute their own plug-in modules. Eclipse SDK is free and open supply package. It absolutely was one in all the first IDEs to run under GNU Class path and it runs without problems under Iced Tea.

4.2 GUI Start-up

In this figure, Choose File 1 button represents the original source file that needs to be chosen. Choose File 2 represents the modified version of the previous original source file.

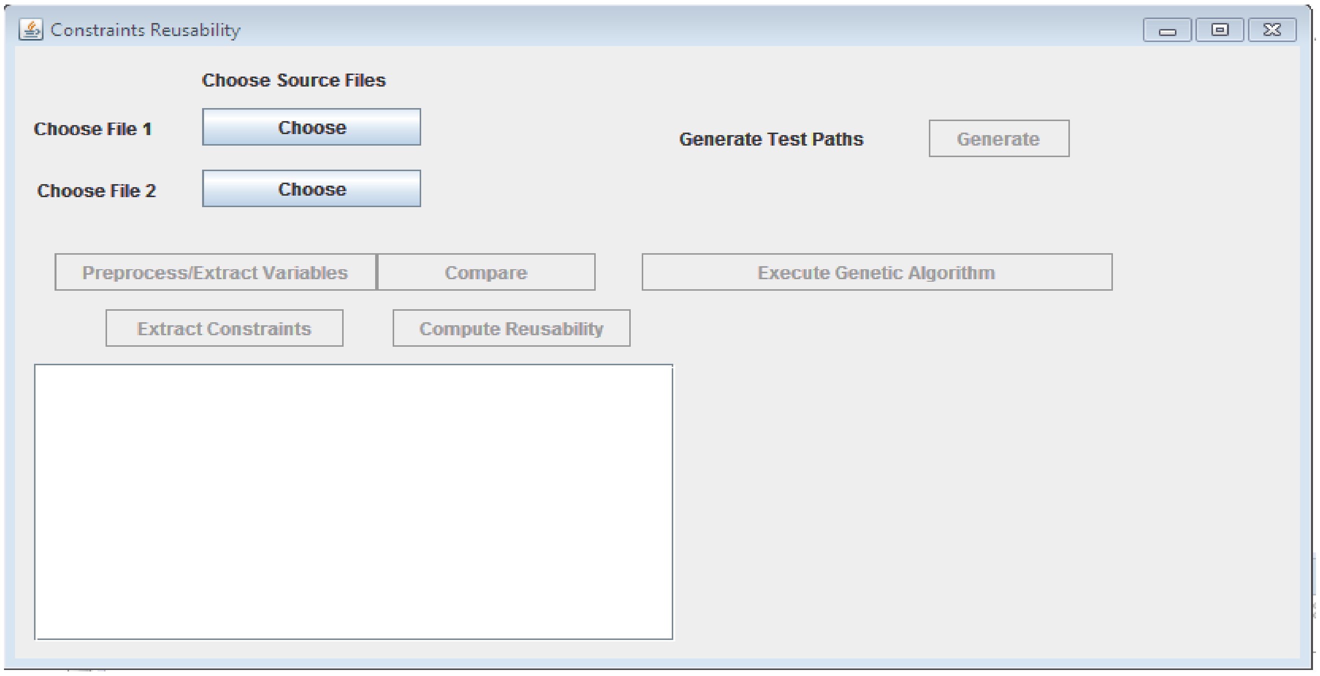


Figure 4.1: GUI at Start-up

4.3 Pop up box for choosing the file

In, this figure the popup box is shown that appears when choose buttons are clicked so that user can select the original and modified version of source file from any location within the computer.

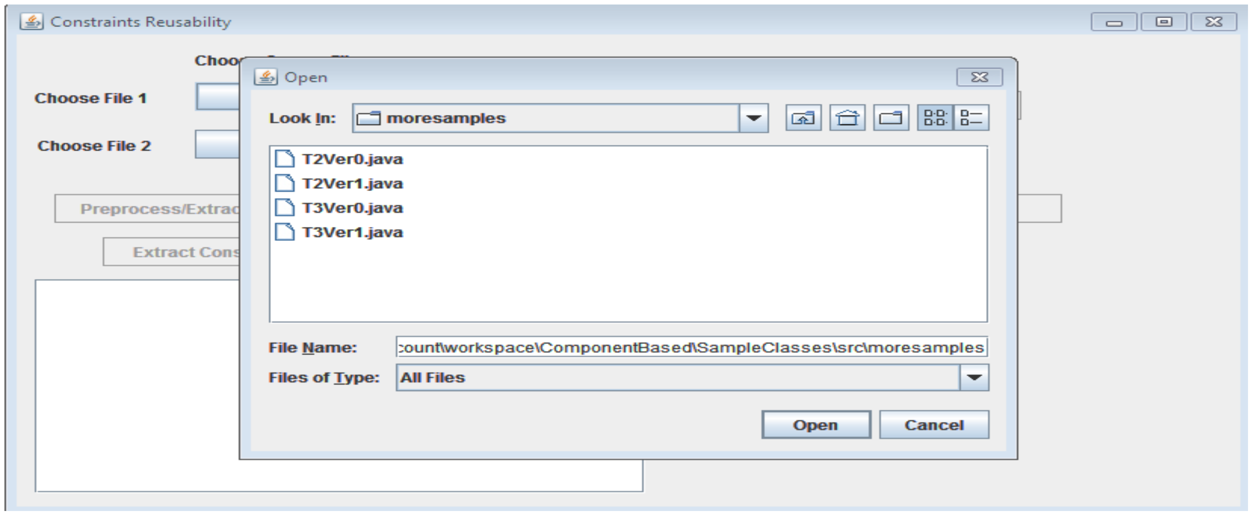
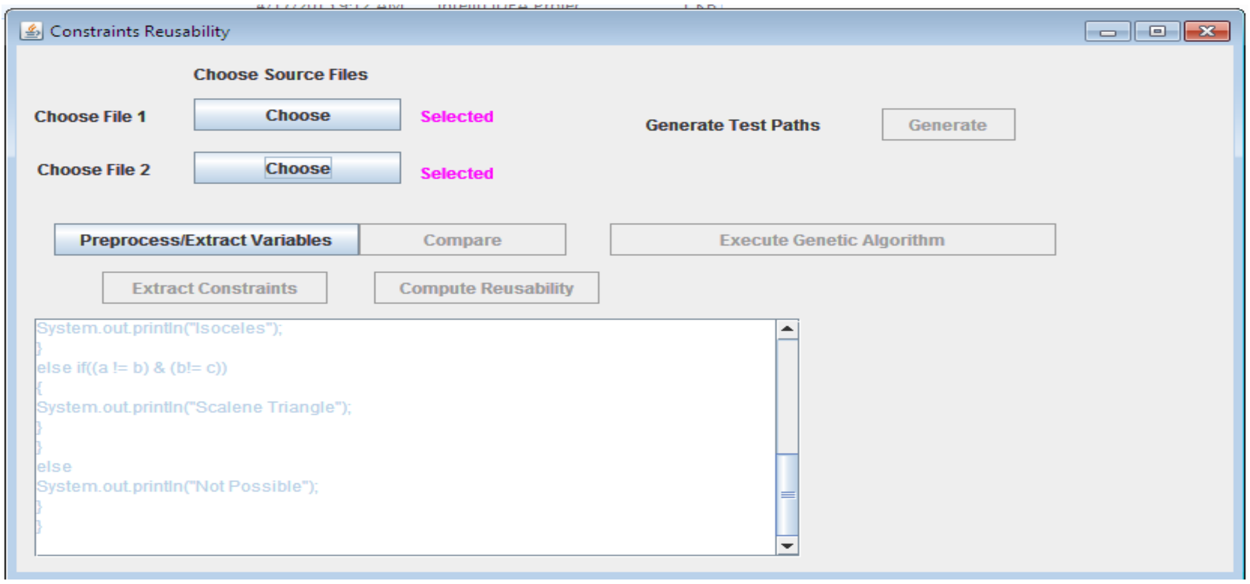


Figure 4.2: Pop up box for choosing the file.

4.4 Output when both files are chosen

The text area in the figure represents the source code of the selected files that is the original source file and modified file.

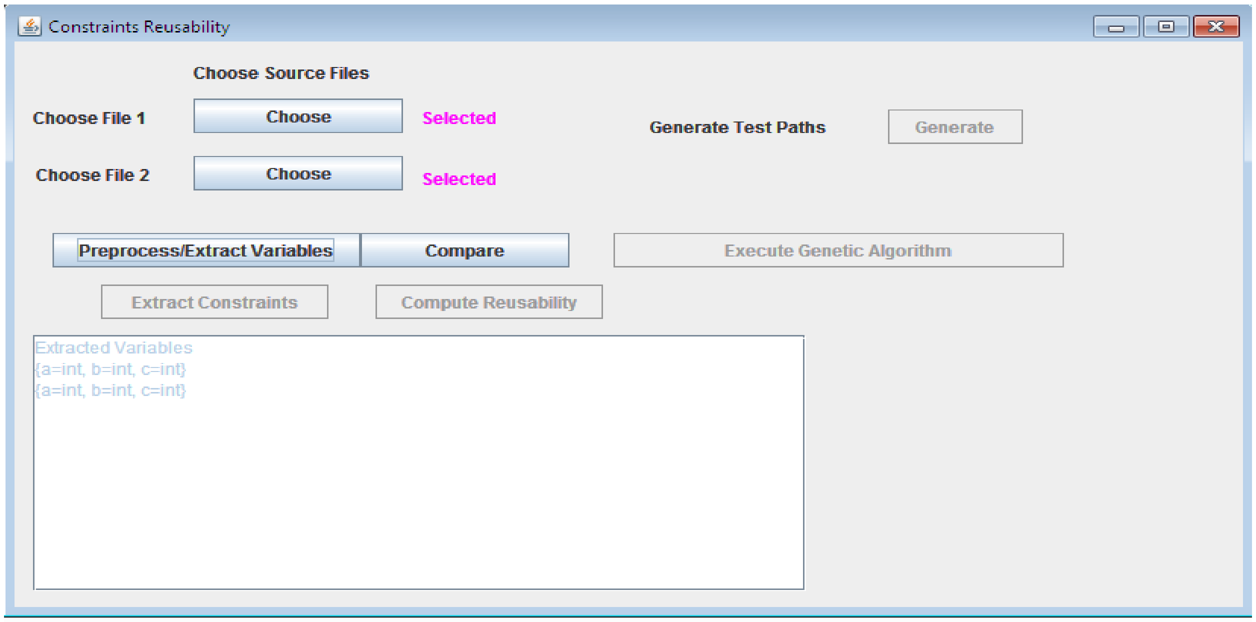


```
System.out.println("Isoceles");
}
else if((a != b) & (b!= c))
{
System.out.println("Scalene Triangle");
}
}
else
System.out.println("Not Possible");
}
}
```

Figure 4.3: Output when both files are chosen

4.5 Output when Pre-process/Extract Variable Button is clicked.

The text area in the figure represents the output of the selected files that is the original source file and modified file. Pre-process/Extract variables function extracts the variables from the pair of files selected.

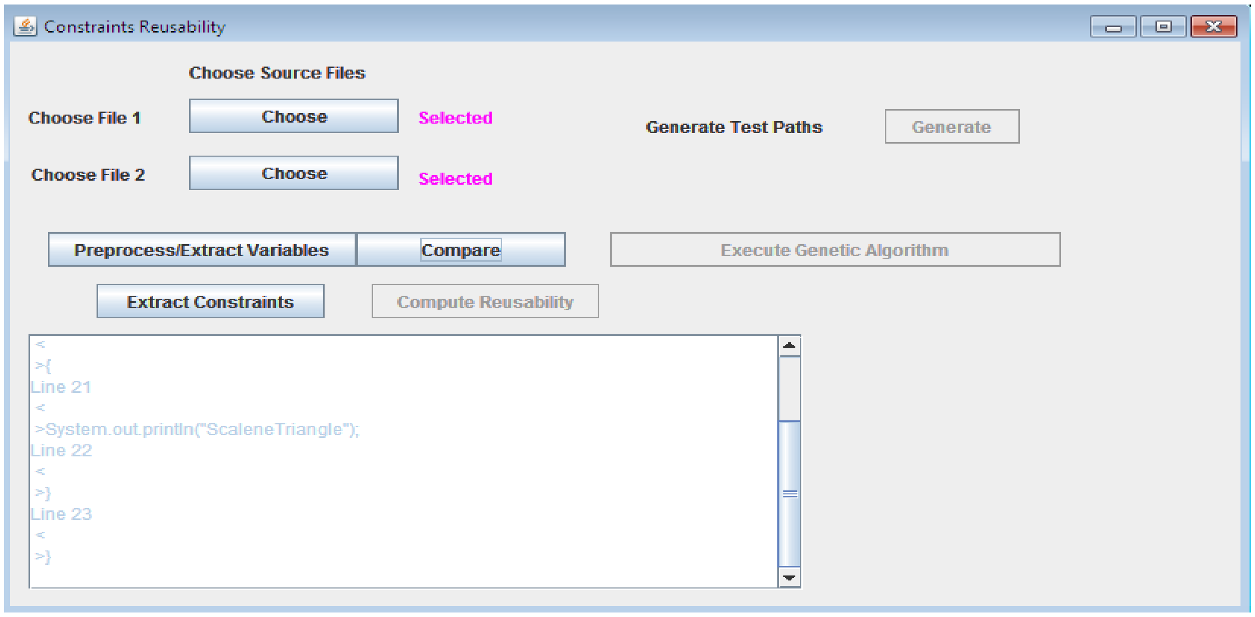


```
Extracted Variables
{a=int, b=int, c=int}
{a=int, b=int, c=int}
```

Figure 4.4: Output when Pre-process/Extract Variable Button is clicked.

4.6 Output when Compare Button is clicked.

The text area in the figure represents the output of the comparison of both the selected files that is the original source file and modified file. Comparison of both files is done in order to find out where actually the changes have been made in the code.

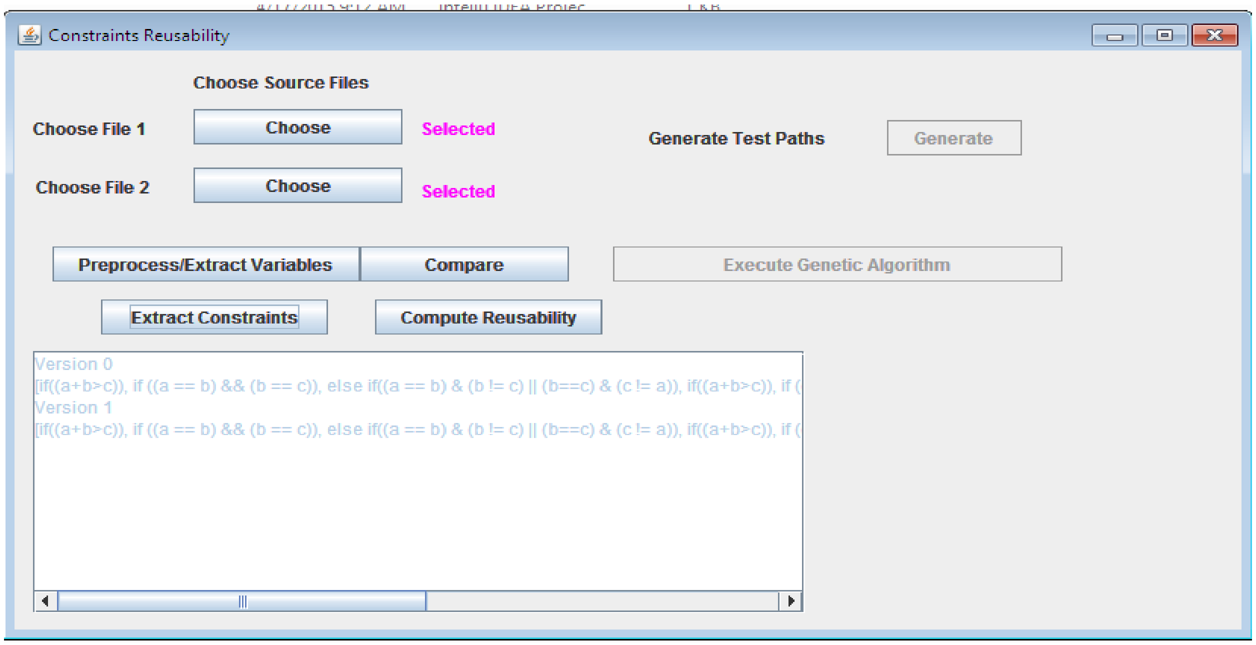


```
<
>{
Line 21
<
>System.out.println("ScaleneTriangle");
Line 22
<
>}
Line 23
<
>}
```

Figure 4.5: Output when Compare Button is clicked.

4.7 Output when Extract Constraints Button is clicked

The text area in the figure represents the constraints of the selected files that is the original source file and modified file. Checksim algorithm is used to find out the constraints and their dependability.

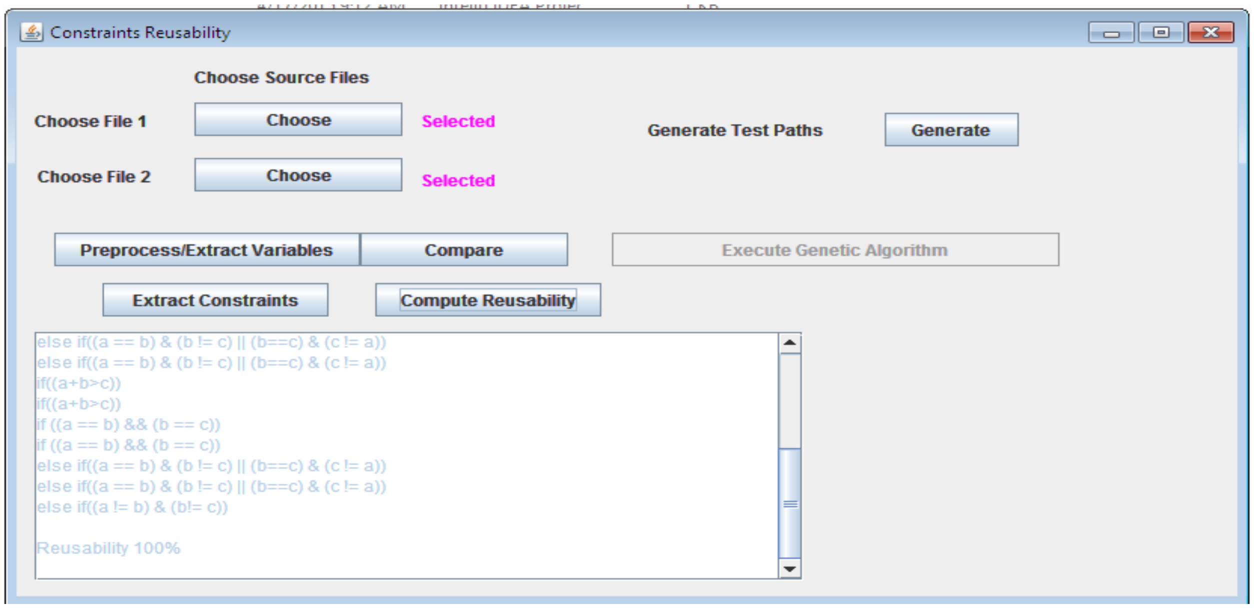


```
Version 0
[if((a+b>c)), if ((a == b) && (b == c)), else if((a == b) & (b != c) || (b==c) & (c != a)), if((a+b>c)), if (
Version 1
[if((a+b>c)), if ((a == b) && (b == c)), else if((a == b) & (b != c) || (b==c) & (c != a)), if((a+b>c)), if (
```

.Figure 4. 6: Output when Extract Constraints Button is clicked.

4.8 Output when Compute Reusability Button is clicked.

The text area in the figure represents the percentage of reusability of the selected files that is the original source file and modified file.



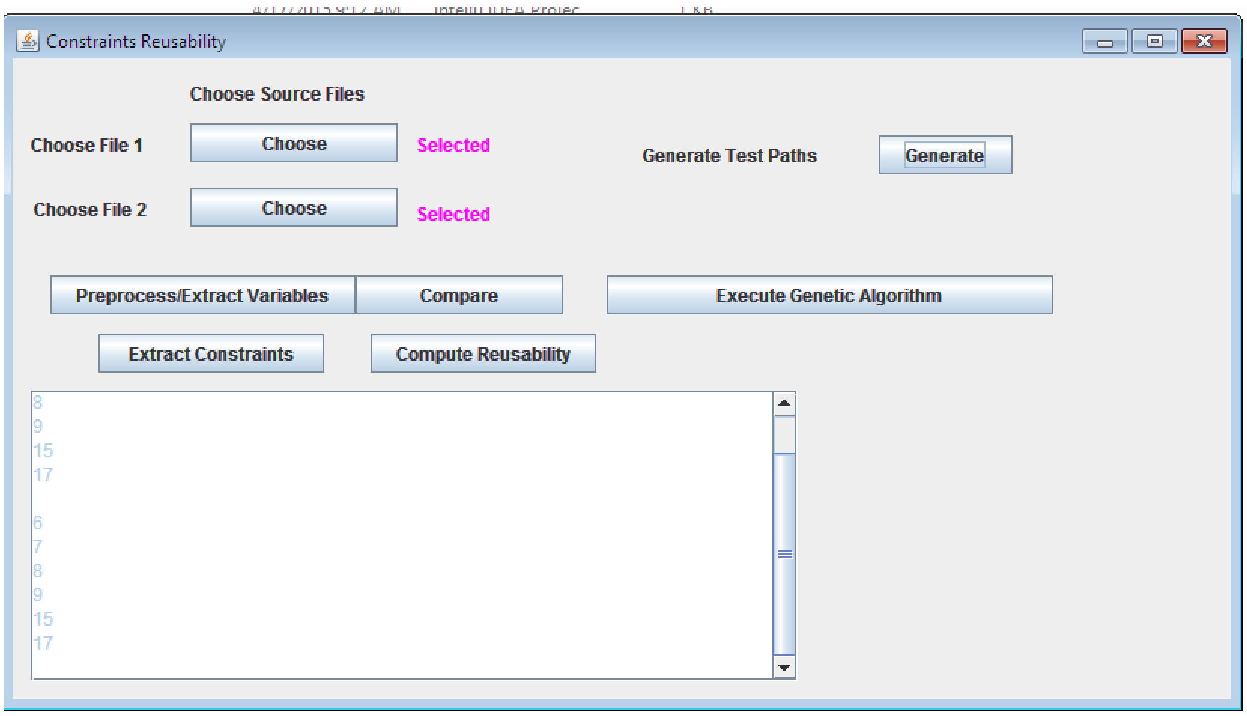
```
else if((a == b) & (b != c) || (b==c) & (c != a))
else if((a == b) & (b != c) || (b==c) & (c != a))
if((a+b>c))
if((a+b>c))
if ((a == b) && (b == c))
if ((a == b) && (b == c))
else if((a == b) & (b != c) || (b==c) & (c != a))
else if((a == b) & (b != c) || (b==c) & (c != a))
else if((a != b) & (b!= c))

Reusability 100%
```

Figure 4.7: Output when Compute Reusability Button is clicked.

4.9 Output when Generate Button is clicked.

The text area in the figure represents the test paths of the selected files that is the original source file and modified file.



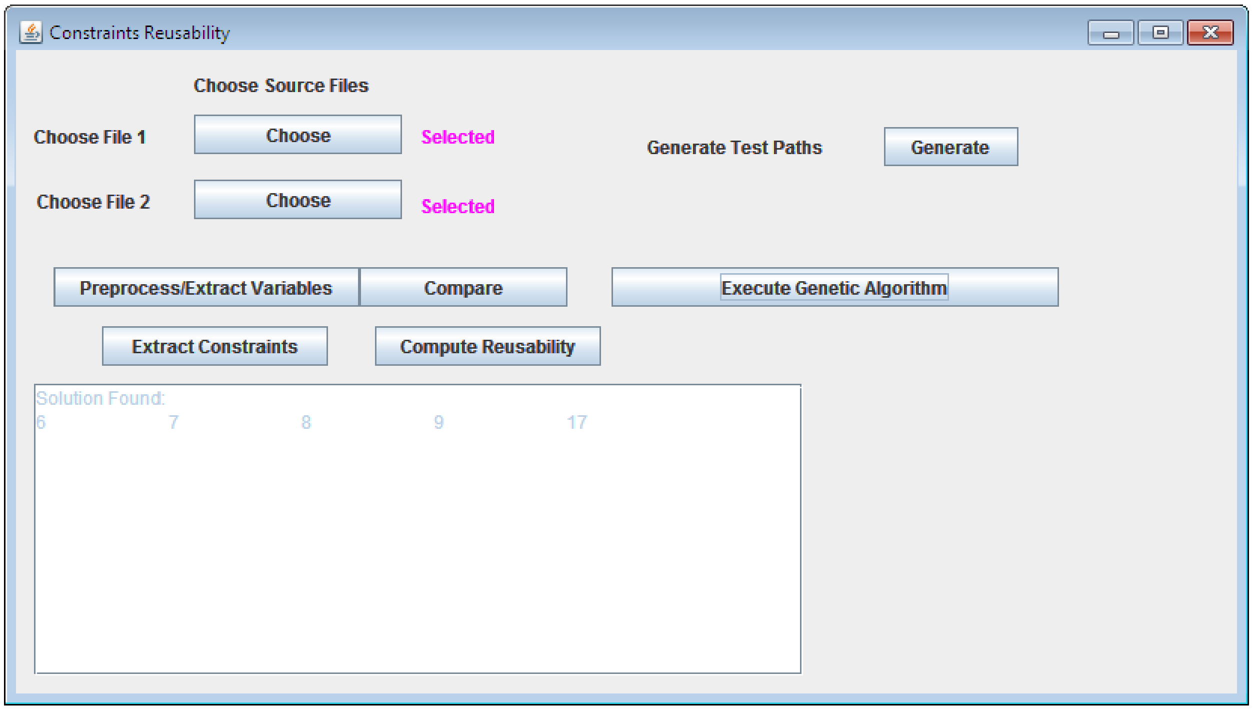
```
8
9
15
17

6
7
8
9
15
17
```

Figure 4.8: Output when Generate Button is clicked.

4. 10 Output when Execute Genetic Algorithm Button is clicked.

The text area in the figure represents the output of paths when Genetic Algorithm is applied on the previous generated paths of the selected files that is the original source file and modified file.



```
Solution Found:
6          7          8          9          17
```

Figure 4. 9: Output when Execute Genetic Algorithm Button is clicked.

Table 4.11 Reusability 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%

Table 1: Result Table

4.12 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.

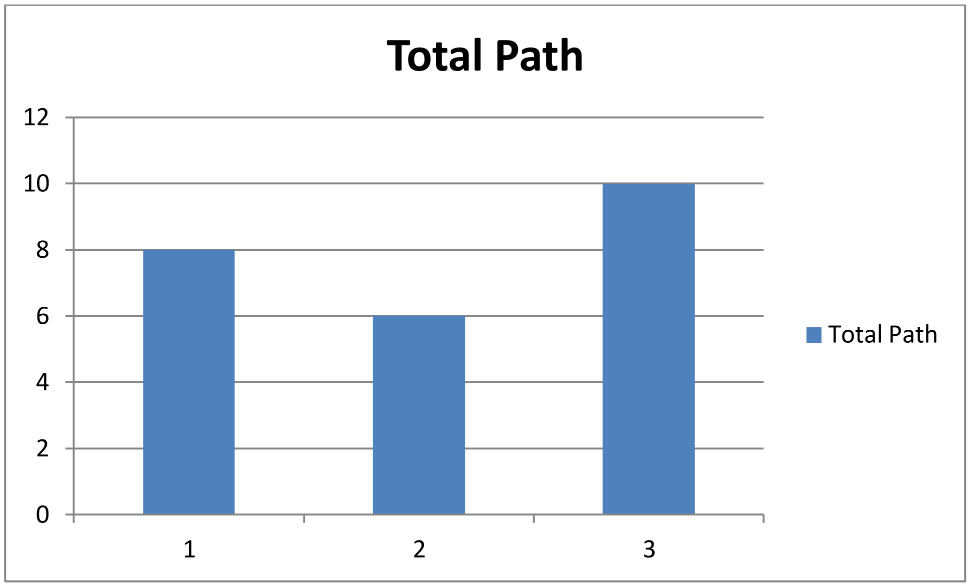


Figure 4.10: Total Number of Paths

4.13: 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

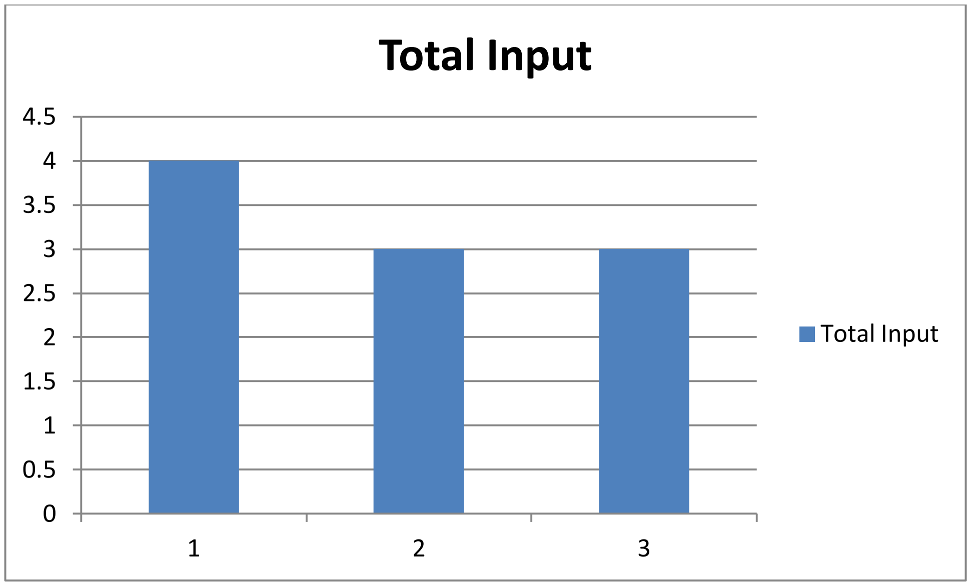


Figure 4.11: Total Number of Inputs

4.14 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.

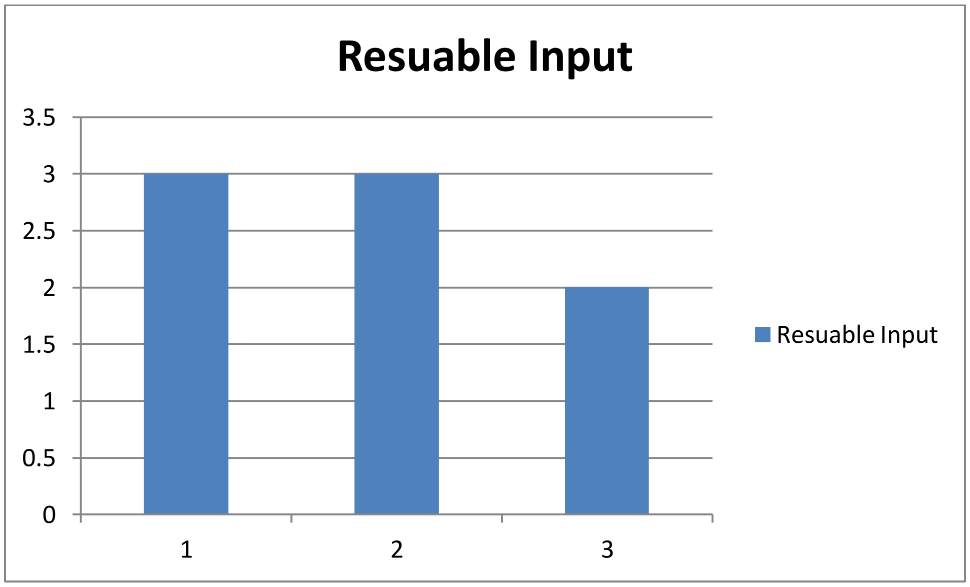


Figure 4.12: Reusable Inputs

4.15 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.

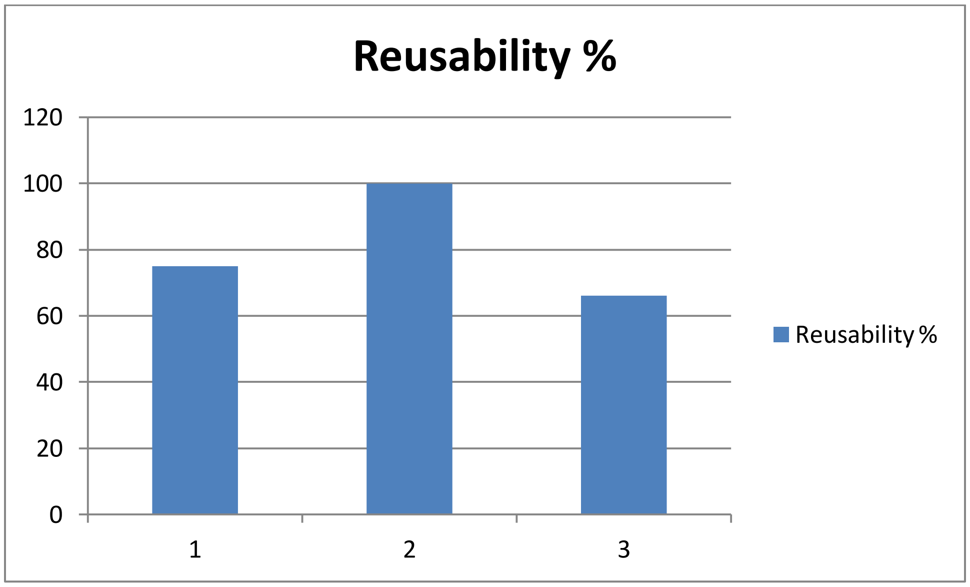


Figure 4.13: Reusability

4.16 Reusability if 1st File

This graph shows that 75% reusability is achieved in terms of input value reusability and constraints reusability

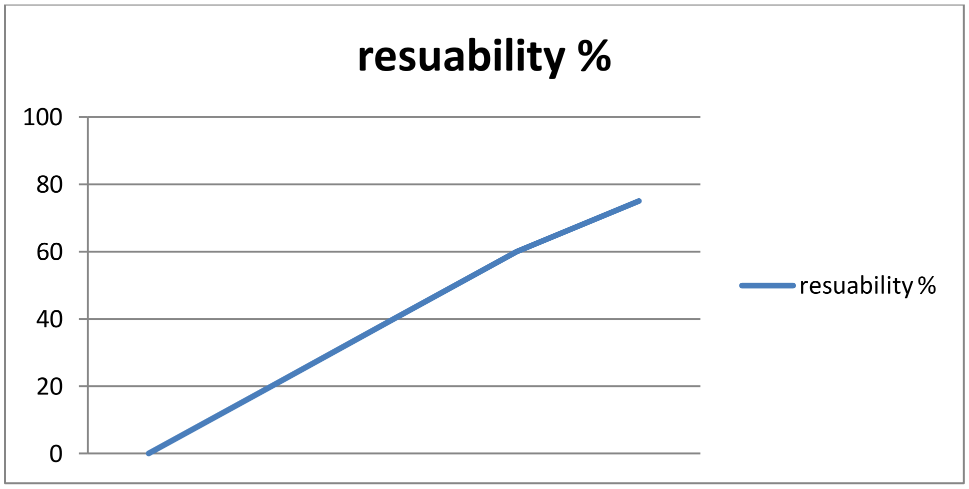


Figure 4.14: Reusability if 1st File

4.17: Reusability of 2nd File

This graph shows that 100% reusability is achieved in terms of input value reusability and constraints reusability

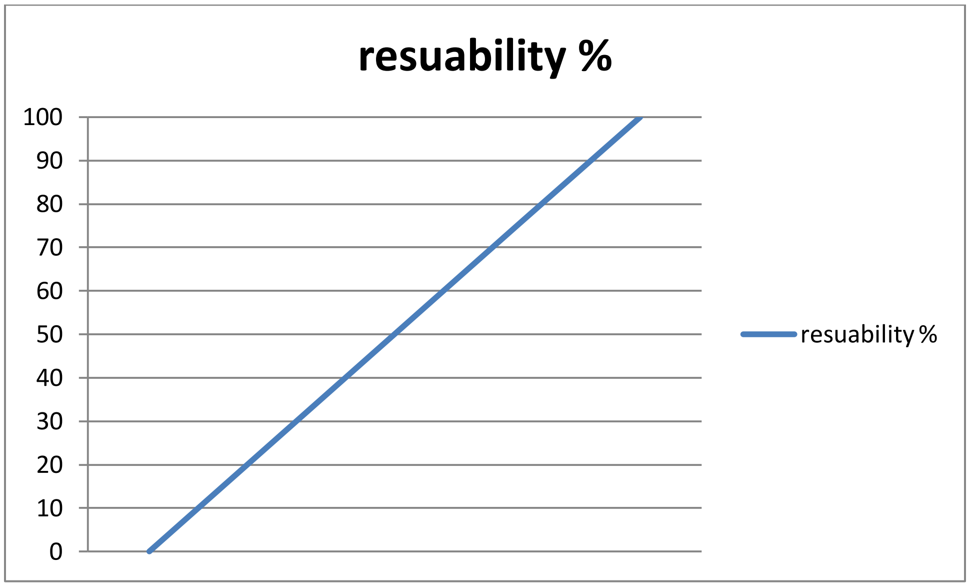


Figure 4.15: Reusability of 2nd File

4.18 Reusability of 3rd File

This graph shows that 66% reusability is achieved in terms of input value reusability and constraints reusability

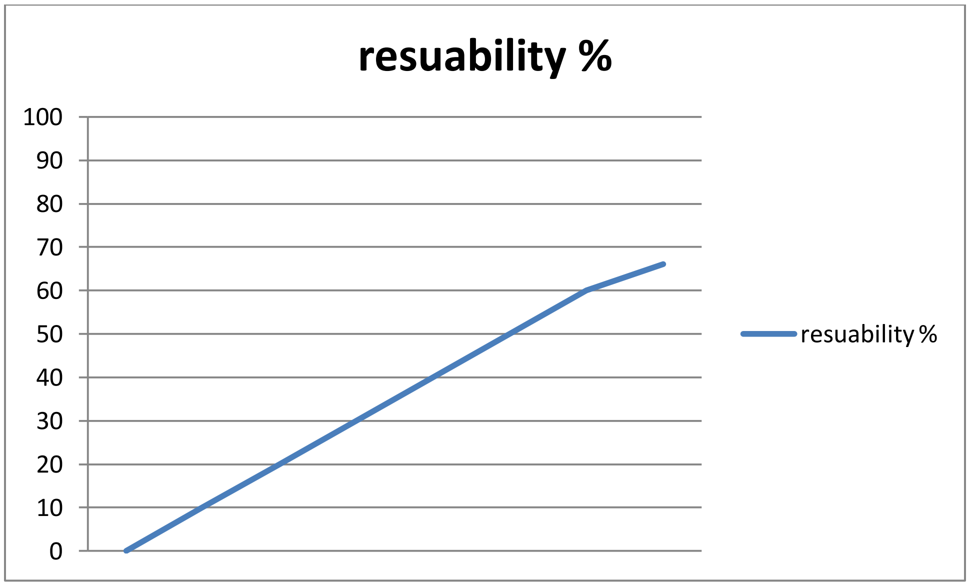


Figure 4.16: Reusability of 3rd File

4.19 Previous Flow of Work

In existing work, the test cases of previous version are created. Then when the code is modified, new test cases are generated. Test Cases from both previous version and Updated version are reused. Reusability is computed.

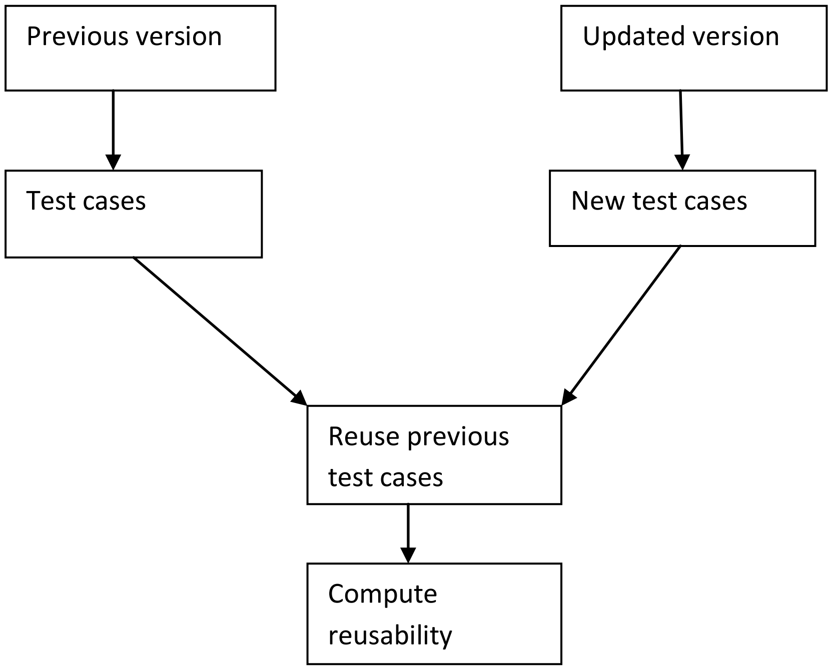


Figure 4.17: flow chart of previous work

4.20 Proposed Work of flow

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.

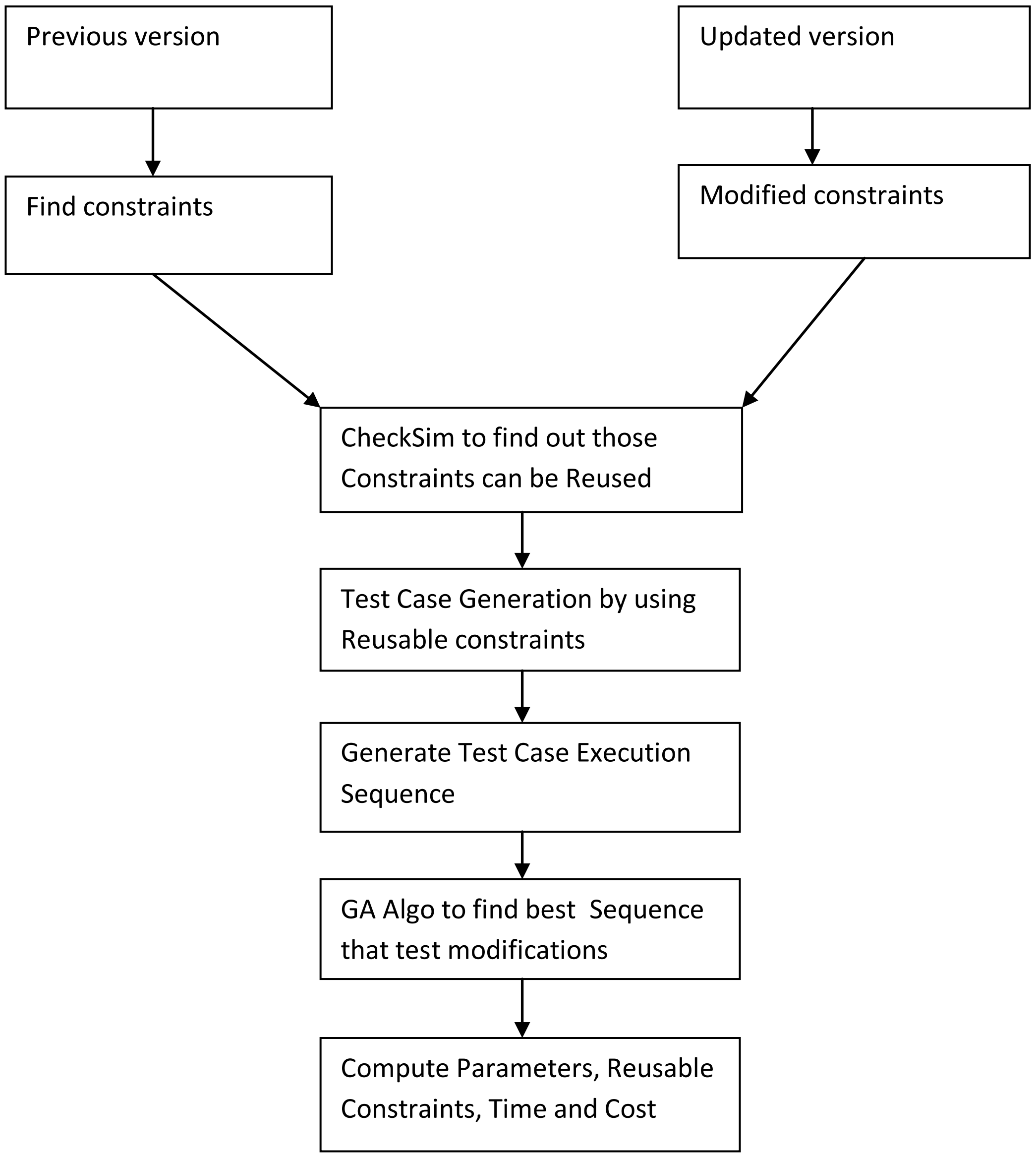

Previous version
Updated version
Find constraints
Modified constraints
CheckSim to find out those Constraints can be Reused
Test Case Generation by using Reusable constraints
Generate Test Case Execution Sequence
GA Algo to find best Sequence that test modifications
Compute Parameters, Reusable Constraints, Time and Cost

Figure 4.18:flow chart of proposed work

CHAPTER 5

CONCLUSION AND FUTURE WORK

Software engineering is the branch of computers to develop platform for various applications to be execute. Various applications that have been used for various purposes have to be tested for their validity and integrity. Due to changes occurred in an organization and requirements of the users different modifications has to be done in software application.

In the process of modification in application has to be validating by using test cases. To validate these modification regression testing has to be performed on each single iteration of the modified version. To perform regression testing each time new test cases has to be designed that use time and cost for a developer. In the purposed work to decrease the efforts and time to create test cases each time reusability of these test cases constraint has been purposes. In the purposed work two application versions have been used that is one is previous version and other is modified version. These modifications have to undergo regression testing for performance evolution. To perform this work the constraints of the previous version and modified version have been matched and selected that can be reused in the generation of test cases.

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. These operators used to evaluate fitness of each test case and on the basis of fitness best sequence has been selected executed.

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.

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.

CHAPTER 6

LIST OF REFERENCES

[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] Hossain, M "Regression Testing for Web Applications Using Reusable Constraint Values" ISSN 14351440, pp 312 – 321, IEEE, 2014.

[4] 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

[5] McMaster, S "Developing a Feedback-Driven Automated Testing Tool for Web Applications" ISSN 978-1-4673-2857-9, pp 210 – 213, IEEE, 2012.

[6[Leotta, M. "Capture-replay vs. programmable web testing: An empirical assessment during test case evolution", ISSN 13916982, ISSN 13916982, pp. 272 – 281, IEEE, 2013.

[7] Lei Xu "A framework for Web applications tests", ISSN 0-7695-2140-1, pp. 300 – 305, IEEE 2014.

[8] N. Prakash1 "Modular Based Multiple Test Case Prioritization N. Prakash1," International Conf. on Computational Intelligence & Computing Research (ICCIC), 2012, pp 1 – 7.

[9]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.

[10] 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.

[11] 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

[12] Arafeen, M.J. "Adaptive Regression Testing Strategy: An Empirical Study" IEEE Conf. on Software Reliability Engineering (ISSRE), 2011, pp 130 – 139.

[13] Harrold, M.J "Empirical studies of a prediction model for regression test selection" IEEE Conf. on Software Engineering, IEEE Transactions, 2002, pp 248 – 263.

[14] Liang You ; Wuhan, China ; Yansheng Lu "A genetic algorithm for the time-aware regression testing reduction problem" published in 8th International Conference on Natural Computation (ICNC), PP 596-599, 2012.

[15] Xiao Qu ; Acharya, M. ; Robinson, B "Configuration selection using code change impact analysis for regression testing" published in 28th IEEE International Conference on software maintenance (ICSM), PP 129-138, 2012.

[16] Engström, E. ; Runeson, P. ; Ljung, A. "Improving Regression Testing Transparency and Efficiency with History-Based Prioritization -- An Industrial Case Study" published in IEEE 4th International Conference on software testing verification and validation (ICST), PP 367-376, 2011.

[17] Shiming Sun ; Xiuping Hou ; Can Gao ; Linlin Sun "Research on optimization scheme of regression testing" published in IEEE 9th International Conference on natural computation (ICNC), PP 1628-1632, 2013.

[18] Kayes, M.I. ; Quality Assurance Eng., Software People, Dhaka, Bangladesh "Test case prioritization for regression testing based on fault dependency" published in IEEE 3rd International Conference on Electronics computer technology (ICECT), PP 48-52, 2011.