DISSERTATION

On

A Paradigm for testing Web Application

SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENT

FOR THE AWARD OF DEGREE

Of

MASTER OF ENGINEERING

(Computer Technology and Application)

Delhi University, Delhi

Submitted By:

RUCHI GOEL

University Roll No

10073

Under the Guidance of:

Dr. Akshi Kumar

Assistant Professor

Department Of Computer Science and Engineering

Delhi College of Engineering, Delhi

DEPARTMENT OF COMPUTER ENGINEERING

DELHI COLLEGE OF ENGINEERING

DELHI UNIVERSITY

2011

CERTIFICATE

I hereby certify that the work is being presented in the thesis report entitled, "**A Paradigm for Testing Web Application**", submitted in partial fulfilment of the requirements for the award of degree of Master of Engineering in Computer Technology & Application at Delhi College of Engineering, Delhi, is a authentic record of my own work carried out under the supervision of **Dr**. **Akshi Kumar** and refers other researcher's works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of any university.

(Ruchi Goel)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Dr. Akshi Kumar
Assistant Professor
Department of Computer Science and Engineering
Delhi College of Engineering, Delhi - 110042

ACKNOWLEDGEMENT

No volume of words is enough to express my gratitude towards my guide **Dr. Akshi Kumar**, Assistant Professor, Computer Science and Engineering, Delhi College of Engineering, Delhi, who has been very concerned and has aided for all the materials essential for the preparation of this thesis report. She has helped me to explore this vast topic in an organized manner and provided me with all the ideas on how to work towards a research-oriented venture.

I am also thankful to **Dr. (Mrs) Daya Gupta**, Head of the Department, Computer Science and Engineering, Delhi College of Engineering, Delhi for the motivation and inspiration that triggered me for the thesis work.

I would also like to thank the staff members and my colleagues who were always there at the need of the hour and provided with all the help and facilities, which I required for the completion of my thesis work.

Most importantly, I would like to thank my husband ,parents and the almighty for showing me the right direction out of the blue, to help me stay calm in the oddest of the times and keep moving even at times when there was no hope

Ruchi Goel
M.E. (Computer Technology and Application)
Department of Computer Science and Engineering
Delhi College of Engineering Delhi-42

ABSTRACT

For any work of literature, a fundamental issue is to identify the individual(s) who wrote it, and conversely, to identify all of the works that belong to a given individual or to identify the individual who writes many papers on same topic.

A web application is an application that can be accessed via a web browser over a network. Web applications contain client side code and server side code. Web applications undergo changes in the maintenance phase, and retesting changed programs is done thereafter. To retest a program after changes, we can select a subset of the whole test suite on the condition that the selected subset will give confidence about covering the changes. As an important method to ensure quality of web applications, Regression Testing techniques are required for adequate selection of these subsets of test cases. This project compares regression testing techniques for web applications based on a safe approach that covers changed elements and other potentially affected ones.

In this project we have implemented a new technique to select test cases for regression testing on web applications based on the Edge Test Tree Graph (ETT) as presented in the paper "A Practical Web Testing Model for Web Application Testing" by Zhongsheng Qian, Huakiou Miao, Hongwei Zeng., School of Computer Engineering and Science, Shanghai University, China[5]. This work proposes a Web Testing Model for web application testing. It starts from constructing the ETT (Event Test Tree) of web application. An algorithm is designed to derive an ETT (Event Test Tree) from the EDG from ETT, we extract the path expressions to generate test paths. Then a test specification is made in C++ to generate the required test cases. Then the web application is modified and Regression Testing is applied. The regression test selection technique is based on identifying changed and potentially changed components. Empirical results show that this technique selects a reduced number of test cases that experience only affected and potentially affected components.we have also studied paper by Abbas Tarhini, Zahi Ismail, Nashat Mansour, Regression Testing Web Applications, 2008 International Conference on Advanced Computer Theory and Engineering[3].

Contents

LIST OF ABBREVIATION

EDG	Event Dependency Graph
ETT	Event Test Tree
TE	Test Expression
API	Application Programming Interface
CGI	Common Gateway Interface
UML	Unified Modeling Language
XML	Extensible Markup Language
SDG	System Dependent Graph
SQL	Sequential Query Language

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter we will expand why we had chosen this work as my major thesis, how we motivated to do this work along with some issues regarding the work done earlier on the same and how we rectify these issues in my thesis with some assumptions given in the problem statement along with some objectives of the thesis and finally we will discuss how we accomplished the objectives in the contribution.

1.1 Motivation

Web applications generate web pages, which contains different kinds of information such as text, images and forms. Web applications provide thousand and thousand of applications in our day to day life.Web application can be simple login page on web site or it can be as complex as word processor or spreadsheet. The basic aim of using web application is that they require very little or no disk space on the client. Web application contains code of server side as well as client side. Now a day's commonly used web applications are login, online shopping, online retail sales, online marketing and many other functions. Every software product has a target Audience for e.g. the audience for online auctions software is different from banking software. So when an organization develops a software their main aim is whether they will meet all the requirements of customers or not. The main aim in testing is to detect errors so that errors can be recovered. Regression testing is to test modified software to ensure that changes are correct and do not adversely affect other parts of the softwareSuppose that you've tested a system completely and we found no errors now assume that we do some changes in our system or in one part and we want to be sure that this change in system will not affect my system means will not introduce any new errors. Testing is to make sure the software hasn't taken a step backwards or "regressed", is called **"regression testing"**.The main aim of Regression testing is that whenever code is modified or changed it is able to detect unexpected faults. Regression testing should be used to check the code's integrity.

Mainly, regression testing is performed during automated builds tigthly to ensure that errors are detected and recovered as soon as possible.

1.2 Problem Statement

User demands are changing day by day and web applications have rapid developing speed keeping all this in mind, regression testing is much important. Since the changed demands result in different versions of Web applications, and the faults usually hiding in the adjusted contents, the regression testing must cover all the related pages. Firstly, we analyze the possible changes in the Web applications and the influences produced by these changes, discussing in the direct-dependent and indirect-dependent way. Very limited work is proposed on regression testing web based Application.

For our work we have studied the paper by Abbas Tarhini[3] tested web application using Event Dependency Graph, in which they compare original and modified web application using affected and potentially affected nodes and reduce test cases. The disadvantage of this approach is that graph includes some cyclic redundancies which can be avoided by using tree.

For our problem statement we have make event dependency graph of original and modified web application. Then we convert Event dependency graph to Event Test Tree for the original and modified system and then identifying the changed nodes in order to select a reduced set of test cases. The results obtained are compared with the results obtained from the Event Dependency Graph technique to devise the best method. The purpose of building the web application to be tested is to provide online facilities for the students pursuing higher education.

1.3 Proposed Work

In this thesis, we provide a review of existing approaches to problems similar to what we outlined in the previous section. We will use regression testing technique to test web application. using Regression testing we compare original and modified web application and check nodes which are affected and potentially affected nodes.When we do regression testing there are two options first is to rerun every existing test case which require lots of time and resources but the problem is we don’t have unlimited time and resources and second is to rerun a subset of existing test cases but it is also not practical to identify subset of test cases from existing test suite, so we propose a model to test web application using Regression testing technique.

First step is to define all the requirements after defining the problem. Requirement modelling provide multiple levels of detail and support for verification, validation and testing prior to implementation. Regression testing means testing a system using a test set T and then again testing the modified system using same test set T’ and check nodes which are affected means which are changed or modified. In short Regression testing means testing the modified system with same test cases which were used previously to test the original system.

Important question is which existing test cases should be used for regression testing? There are two options foe that first is that we rerun every test case and second option is that we rerun subset of existing test cases.The events represent changes in the system state, and services define the states in which they can be running, and how the system reacts accordingly

.

The regression testing technique is summarized by the following steps:

1) Model the web application and the modified web application by using **Event Dependency Graph**

2) Convert **Event driven graph to Event Test Tree** which will avoid scalability issues for original and modified web application

3) Identify the changed nodes by comparing the nodes from both tree

4) Identify the potentially affected nodes

5) Select the test cases that pass through the changed nodes and the potentially affected nodes.

6) Calculate the reduction in test cases from original and modified one

1.4 Organization of thesis

In the above chapter, we had discussed the motivation, problem statement, objective of the thesis and finally the contribution. Chapter 2 provides the literature review of related works. Chapter 3 has the complete details design of the new approach, their algorithms,. Chapter 4 shows the experimental setup and results of the proposed system and finally chapter 5 consists of the conclusion and possible future work or directions in this area and it finally ends with the links and references (bibliography) and appendices.

1.5 Chapter Summary

In this chapter we had discussed the motivation of the author to do this work and it also includes some issues regarding the previous work done on the same. Finally author had described the objectives of the thesis and how he accomplishes these objectives in his research using some assumptions and Event Dependency Graph and Event Test Tree.

Chapter 2

Literature Review

In this chapter, we will provide a literature review for related work in this area. The work done on regression testing web application is very limited.Although For two reasons, the list of related works is long: one reason is that this task is at the intersection of several important and active machine learning and NLP research areas therefore many different approaches can be adapted to become relevant. The other important reason is that our main approach of using graphical models has gained tremendous interest in recent years due to successful application of these methods. We have made an attempt to organize some of related literature with respect to their relevance to this task. We hope the result of this literature review to help anyone who would like to design a new method for a different task based on what is known about previous methods.

2.1 INTRODUCTION

Models are considered an essential step in capturing different system behaviors and simplifying the analysis required to check or improve the quality of software. Verification and testing of web software requires effective modeling techniques that address the specific challenges of web applications.

Like many software domains, web applications are becoming more complex. This complexity arises due to several factors, such as a larger number of hyperlinks, more complex interaction, and the increased use of distributed servers. Modeling can help to understand these complex systems, and several papers in the literature have studied the specific problem of modeling web applications. In some cases, new models have been proposed, while in other cases, existing modeling techniques have been adapted from other software domains. Modeling can help designers during the design phases by formally defining the requirements, providing multiple levels of detail, and providing support for testing prior to implementation. Support from modeling can also be used in later phases to support validation and verification.

2.2 Web Application

A **web application** is an application that is accessed over a network such as the Internet or an intranet[18]. The main drawback in web application is that it may have several entry points, and users can not prevent himself from these complicated interactions. Web applications are becoming more complex due to increase use of distributed servers, larger number of hyperlinks, and their usage in our daily life. Modelling can help us in engaging in these complex interactions. A web application is a dynamic extension of a web or application server. There are two types of web applications:

- *Presentation-oriented*: A presentation-oriented web application generates interactive web pages containing various types of markup language (HTML, XML, and so on) and dynamic content in response to requests.
- *Service-oriented*: A service-oriented web application implements the endpoint of a web service. Presentation-oriented applications are often clients of service-oriented web applications. Web applications are popular due to the ubiquity of web browsers, and the convenience of using a web browser as a client, sometimes called a thin client. The ability to update and maintain web applications without distributing and installing software on potentially thousands of client computers is a key reason for their popularity, as is the inherent support for cross-platform compatibility. Common web applications include webmail, online retail sales, online auctions, wikis and many other functions. A web application is often structured as a three-tiered application[19]. As shown in Fig. 2.1, the web browser represents the first tier. The web server that implements CGI, PHP, Java Servlets or Active Server Pages (ASP),along with the application server that interacts with the database and other web objects is considered the middle tier. Finally, the database along with the DBMS server forms the third tier.

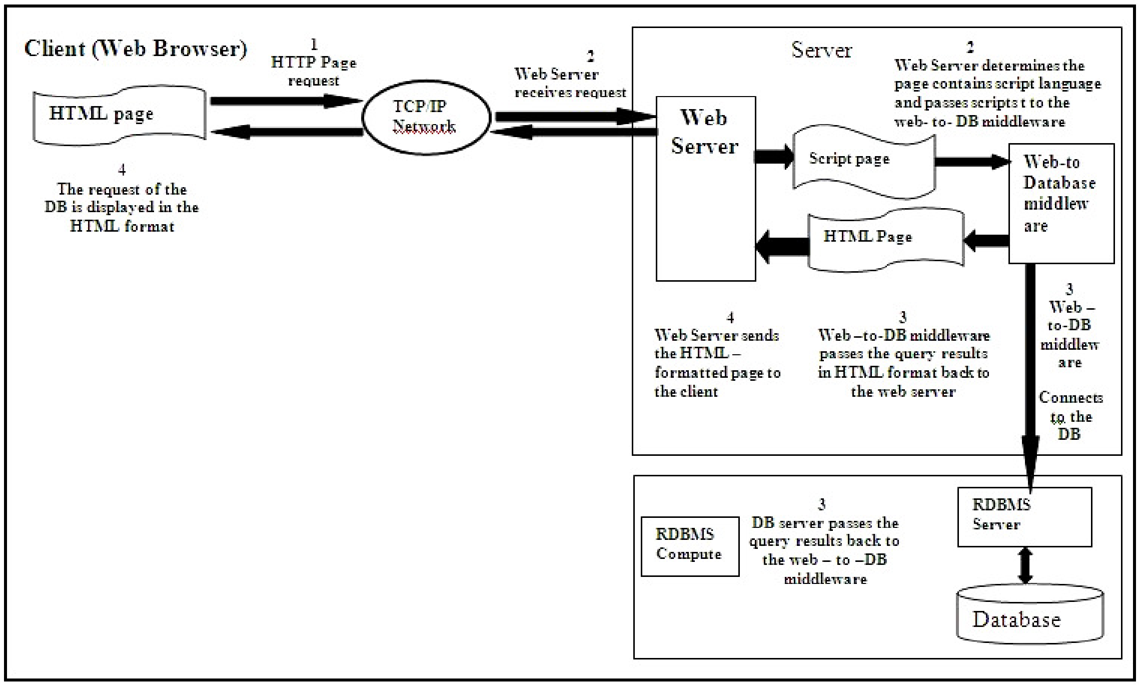

Client (Web Browser)
1 HTTP Page request
HTML page
TCP/IP Network
2 Web Server receives request
4 The request of the DB is displayed in the HTML format
Server
2 Web Server determines the page contains script language and passes scripts t to the web- to- DB middleware
Web Server
Script page
Web-to Database middlew are
HTML Page
4 Web Server sends the HTML – formatted page to the client
3 Web –to-DB middleware passes the query results in HTML format back to the web server
3 Web – to-DB middlew are Connects to the DB
RDBMS Server
RDBMS Compute
3 DB server passes the query results back to the web – to –DB middleware
Database

Figure:2.1 Web Application Components

Web applications generate web pages, comprising different kinds of information such as text, images and forms. These web pages can be either static or dynamic. Static pages reside on a web server and contain only HTML and client side executable code (e.g. JavaScript) and are served by the web server. Dynamic pages are generated as the result of the execution of various scripts and components on the server. These pages contain a mixture of HTML source and executable code, and are served by the application server.Some researchers consider a website to be simply a set of related web pages grouped together by some means on a server, or in a folder on a server. Such pages are static pages that don't use dynamic features and thus need not be processed by the application servers.The advantages and disadvantages of using web applications are;

Benefits

- Web applications do not require any complex "roll out" procedure to deploy in large organizations. A compatible web browser is all that is needed;
- Browser applications typically require little or no disk space on the client;

- They require no upgrade procedure since all new features are implemented on the server and automatically delivered to the users;
- Web applications integrate easily into other server-side web procedures, such as email and searching.
- They also provide cross-platform compatibility in most cases (i.e., Windows, Mac, Linux, etc.) because they operate within a web browser window.

Drawbacks

- Web applications absolutely require compatible web browsers. If a browser vendor decides not to implement a certain feature, or abandons a particular platform or operating system version, this may affect a huge number of users.
- Standards compliance is an issue with any non-typical office document creator, which causes problems when file sharing and collaboration becomes critical.
- Since many web applications are not open source, there is also a loss of flexibility, making users dependent on third-party servers, not allowing customizations on the software and preventing users from running applications offline (in most cases). However, if licensed, proprietary software can be customized and run on the preferred server of the rights owner.

2.2.1. The Phases in a Web Application Project

The Web application development process has 4 phases:

1. Envisioning the nature and direction of the project
2. Devising the plan
3. Development
4. Testing, support and stability

1. Envisioning the nature and direction of the project

In this phase, the management and developers assigned to the project come together and establish the goals that the solution must achieve. This includes recognizing the limitations that are placed on the project, scheduling, and versioning of the application.

By the end of this phase, there should be clear documentation on what the application will achieve.

2. Devising the plan

In this phase, team determine the "hows" of the application. What scripting language is most appropriate, which features must be included, and how long will it take? These are some of the questions that must be answered through this planning phase. The main tangents at this point are the project plan and functional specification. The project plan determines a timeframe of events and tasks, while the functional specification outlines in detail how the application will function and flow.

3. Development

Once the project plan and functional specification are ready, a baseline is set for the development work to begin. The programmer/s or Web developer/s begins coding, testing and publishing data. This phase establishes the data variables, entities and coding procedures that will be used throughout the remainder of the project. A milestone document is prepared by the development team, which is then handed to management for review.

4. Testing, support and stability

The stability phase of the application project mainly focuses on testing and the removal of bugs, discrepancies and network issues that may otherwise cause the application to fail. It is here that policies and procedures are established for a successful support system.

2.2.2. Challenges in Analysis and Modelling Of Web Application

Web applications are evolving rapidly, as many new technologies, languages, and programming models are used to increase the interactivity and the usability of web applications. This inherent complexity brings challenges to modelling, analysis, testing, and verification of this kind of software. Some of these challenges are: the diversity and complexity of the web application environment increases the risk of non-interoperability and the complexity of integration. Web applications interact with many components that run on diverse hardware and software platforms. They are written in diverse languages and they are based on different programming approaches such as procedural, OO, interpreted, and hybrid languages such as Java Server Pages (JSPs). The client side includes browsers, HTML, embedded scripting languages and applets. The server side includes CGI, JSPs, Java Servlets, and .NET technologies. They all interact with diverse back-end engines and other components that are found on the web server or other servers. The integration of such components and the web system in general is extremely loose and dynamically coupled, which provides powerful abstraction capabilities to the developers, but makes analysis for testing and verification extremely difficult.

Another major challenge comes from the dynamic behavior, including dynamically generated client components, dynamic interaction among clients and servers, and the Continual changes in the system context and web technologies.

Web applications may have several entry points, and users can engage in complicated interactions that the web application cannot prevent. Web applications often contain database components and may provide the same data to different users. In these cases, applying access control mechanisms becomes an important requirement for safe and secure access to web application resources, and the process of implementing and applying such rules is considered a great challenge.

2.2.3 Influences Produced by changing web Application

A Web page is consisted of many categories of elements, and the changes to some elements cannot influence others, such as the adjustment for page layout, or the literal changes to the content, etc. These changes only need specification validation and not need to consider the inter-relationships among them. As the changes to some elements may influence other pages, further analysis are needed to check the dependent relationships among the related pages.

These dependent relationships are divided into two categories: direct dependent and indirect-dependent. One page can also be consisted of several pages, i.e., there exists including relationship among these pages. But in fact, the pages are independent (except framework) and these can be disposed as several single pages.

Hyperlink element and form element can connect two pages directly (form can be Considered as a special hyperlink), i.e., one is the page that contains these elements, and the other is the page that is pointed by these elements. The basic changes are insert hyperlink and delete hyperlink. Thus, when insert a hyperlink, the connections are inspected between the current page and the object page, and the object page can be inside page (inside the website), outside page (outside the website), or static page, dynamic page.

If the object page is outside the website, only check whether it is reachable. If it is inside the website, it is considered in the static and dynamic ways. When the static page is considered, only 1 is added to the in-degree value of this page. While the dynamic page is considered, there is still need to validate the function of the form. When a hyperlink is deleted, the in-degree value is adjusted and it is checked whether it produces an isolate page. Furthermore, modify a hyperlink is the combination of the two before, i.e., first delete a hyperlink, and then add a hyperlink.

When the form is researched deeply, it can be found that it produces not only the direct dependent between pages but also the indirect-dependent among several pages by hidden transferring data. Otherwise, there may be indirect-dependent between two pages caused by visiting the shared variables. For example, the script programs in the server side can execute the actions such as new, modify, visit, delete an item in the database, and the operations of new, modify, delete are the writing operations to the database while the operation of visit is the reading one. There are dependent relationships between the writing and reading operations. The indirect-dependents can be obtained by analyzing the Definition-usage relationships of variables and the related method is similar to the traditional software analysis. But since the Web applications change frequently, the whole system dependent graph cannot be constructed since this task is too hard and tedious, but adapt the Forward and Backward Search Method to gain the indirect dependents.

2.3 Testing

Testing is one of the most crucial and indispensable part of an effective and efficient project. It is the phase where the errors remaining from all the previous phases must be detected. The basic goal of testing is that it performs a very critical role for quality assurance and for ensuring the reliability of software.

During testing, the program to be tested is executed with a set of test cases, and the output of the program for the test cases is evaluated to determine if the program is performing as expected. Thus, testing forms the first step in determining the errors in a program. Testing a large system is a complex activity, and like any complex activity it has to be into smaller activities. Due to this, for a project, incremental testing is generally performed, in which components and subsystems of the system are tested separately before them to form the system for system testing.

2.3.1 Testing Fundamentals

The theoretical foundations of testing are as follows:

(a) Error: It refers to the discrepancy between a computed,observed,or measured value and the true, specified, or theoretically correct value.
Error is also used to refer to human action those results in software containing a defect or fault. This definition is quite general and encompasses all the phases.

(b) Fault: It is a condition that causes a system to fail in performing its required function. A fault is the basic reason for software malfunction and is synonymous with the commonly term bug.

(c) Failure: It is the inability of a system or component to perform a required function according to its specifications. A software failure occurs if the behavior of the software is different from the specified behavior. Failures may be caused due to functional or performance reasons.

(d)Test Cases: Test cases basically reveal the presence of faults which is central to successful testing. Each test case costs money and thus, efforts must be made to minimize the number of test cases.

(e)Test Oracles: To test any program, we need to have a description of its expected behaviour and a method of determining whether the observed behaviour conforms to the expected behaviour. For this we need a test oracle.

A test oracle is a mechanism, different from the program itself that can be used to check the correctness of the output of the program for the test cases. It can be depicted diagrammatically as:

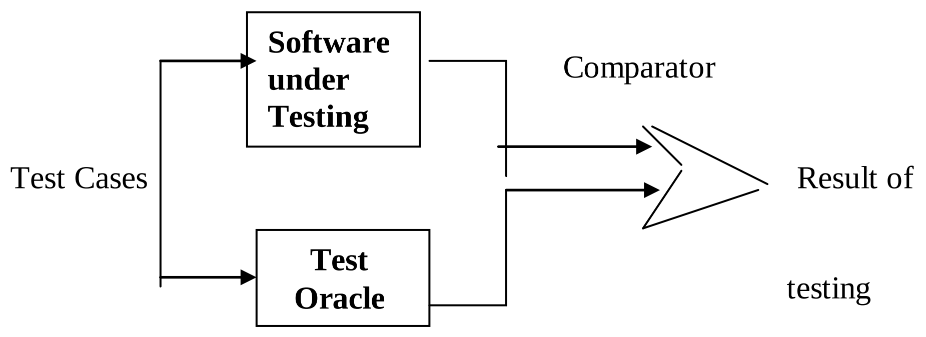


Figure 2.2 Testing Process

2.3.2. Different Approaches to Testing

(a) **Top-Down and Bottom-Up Approaches :**

For this, we assume that a system is a hierarchy of modules. In top-down strategy, we start by testing the top of the hierarchy, and we incrementally add modules that it calls and then test the new combined system. This approach requires stubs to be written. A stub is a dummy routine that simulates a module.

The bottom-up approach starts from the bottom of the hierarchy. First the modules at the very bottom, which have no subordinates, are tested. Then these modules are combined with higher-level modules for testing. To perform this approach, drivers are needed to set up the appropriate environment and invoke the module. It is the job of the driver to invoke the module under testing with the different set of test cases.

Both the above approaches are incremental, starting with testing single modules and then adding untested modules to those that have been tested, until the entire system is tested.

(b) Functional and Structural Approaches :

In functional testing, the structure of the program is not considered. Test cases are decided solely on the basis of the requirements or specifications of the program or module, and the internals of the module or the program are not considered for selection of test cases. Due to its nature, this testing is also often called "black box testing".

Structural testing, on the other hand, is concerned with testing the implementation of the program. The intent of this testing is not to exercise all the different input or output conditions but to exercise the different programming structures and data structures used in the program. In this approach, test cases are generated based on the actual code of the program or module to be tested. This structural approach is sometimes called "glass box testing" or "white box testing".

2.3.3. Testing Process

As testing is the last phase before the final software is delivered, it has the enormous responsibility of detecting any type of error that may be in the software. As testing is the costliest activity in software development, it is important that it be done efficiently. Due to this, different levels of testing are used in the testing process, each level of testing aims to test different aspects of the system and are represented graphically as :

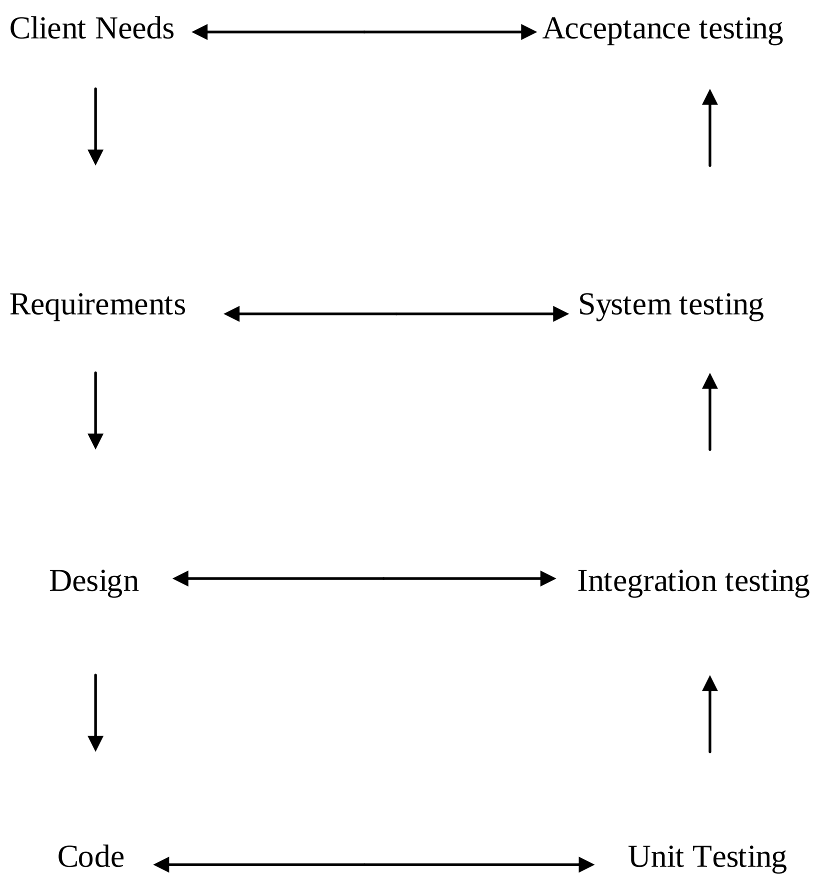


Figure 2.3: Levels Of Testing

The following different levels of testing attempt to detect different types of faults :

a) **Unit Testing :** The first level of testing is called unit testing. In this, different modules are tested against the specifications produced during design for the modules. It is essentially for verification of the code produced during the coding phase, and hence the goal is to test the internal logic of the modules. It is typically done by the programmer of the module. It is also called as module testing.

b) **Integration Testing :** The next level of testing is often called integration testing. In this, many unit tested modules are combined into subsystems, which are then tested. The goal here is to ensure emphasis on testing interfaces between modules. This testing activity can be considered during testing the design.

c) **System & Acceptance Testing :** Here the entire software system is tested. The reference document for this process is the requirement document and the goal is to see if the software meets its requirements. This is essentially a validation exercise, and in many situations it is the only validation activity. Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactorily. Testing here focuses on the external behaviour of the system.

The testing process usually commences with a test plan, which is the basic document guiding the entire testing of the software. It specifies the levels of testing and the units that need to be tested.

2.3.4 TEST CASES

“A test case has components that describes an input, action or event and an expected response, to determine if a feature of an application is working correctly.”

A **test case** in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. The mechanism for determining whether a software program or system has passed or failed such a test is known as a **test oracle**. In some settings, an oracle could be a requirement or use case, while in others it could be a heuristic. It may take many test cases to determine that a software program or system is functioning correctly. Test cases are often referred to as **test scripts**, particularly when written. Written test cases are usually collected into test suites.

In order to fully test that all the requirements of an application are met, there must be *at least two test cases for each requirement*: one positive test and one negative test; unless a requirement has sub-requirements. In that situation, each sub-requirement must have at least two test cases. Keeping track of the link between the requirement and the test is frequently done using a traceability matrix. Written test cases should include a description of the functionality to be tested, and the preparation required to ensure that the test can be conducted.

<u>Formal test cases</u>

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 should test a precondition and the expected output should test a post condition.

<u>Informal test cases</u>

For applications or systems without formal requirements, test cases can be written based on the accepted normal operation of programs of a similar class. In some schools of testing, test cases are not written at all but the activities and results are reported after the tests have been run.

A test case is usually a single step, or occasionally a sequence of steps, to test the correct behaviour/functionalities, features of an application. An expected result or expected outcome is usually given.

Additional information that may be included:

- test case ID
- test case description
- test step or order of execution number
- related requirement(s)
- depth

- test category
- author
- check boxes for whether the test is automatable and has been automated.

Additional fields that may be included and completed when the tests are executed:

- pass/fail
- remarks

Larger test cases may also contain prerequisite states or steps, and descriptions. A written test case should also contain a place for the actual result.

These steps can be stored in a word processor document, spreadsheet, database or other common repository. In a database system, you may also be able to see past test results and who generated the results and the system configuration used to generate those results. These past results would usually be stored in a separate table.

Test suites often also contain

- Test summary
- Configuration

Besides a description of the functionality to be tested, and the preparation required to ensure that the test can be conducted, the most time consuming part in the test case is creating the tests and modifying them when the system changes. Under special circumstances, there could be a need to run the test, produce results, and then a team of experts would evaluate if the results can be considered as a pass. This happens often on new products' performance number determination. The first test is taken as the base line for subsequent test / product release cycles.

Writing effective test cases is a skill and that can be achieved by some experience and in-depth study of the application on which test cases are being written. For any application basically all the types of test cases including functional, negative and boundary value test cases should be covered. **Test** cases should be simple and easy to understand and to the point without including essay like explanations.

2.4 WEB APPLICATION TESTING

2.4.1. Background

Websites impose some entirely new challenges in the world of software quality. Within minutes of going live, a Web application can have many thousands more users than a conventional, non-Web application. The immediacy of the Web creates immediate expectations of quality and rapid application delivery, but the technical complexities of a Website and variances in the browser make testing and quality control that much more difficult, and in some ways, more subtle, than "conventional" client/server or application testing. Automated testing of Websites is an opportunity and a challenge.

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[20]. Issues such as the security of the web application, the basic functionality of the site, its accessibility to handicapped users and fully able users, as well as readiness for expected traffic and number of users and the ability to survive a massive spike in user traffic, both of which are related to load testing

2.4.2. Defining Website Quality and Reliability

Like any complex piece of software there is no single, all inclusive quality measure that fully characterizes a Website (by which we mean any web browser enabled application).

2.4.3. Dimensions of Quality:

There are many dimensions of quality; each measure will pertain to a particular Website in varying degrees. Some common measures are :

- *Timeliness:* Websites change often and rapidly. This measure indicates how much has a Website changed since the last upgrade.

- *Structural Quality:* This indicates how well do all of the parts of the Website hold together. It checks weather all links inside and outside the Website are working or not.
- *Content:* It matches the content of critical pages with the specifications.
- *Accuracy and Consistency:* it checks the data for accuracy and consistency checks are also performed.
- *Response Time and Latency:* Website server response to a browser request within certain performance parameters is evaluated.
- *Performance:* The Browser --> Web --> Website --> Web --> Browser connection link is checked for swiftness.

2.4.4. Impact of Quality:

Quality remains is in the mind of the Website user. A poor quality Website, one with many broken pages and faulty images, with Cgi-Bin error messages, etc., may cost a lot in poor customer relations, lost corporate image, and even in lost sales revenue. Very complex, disorganized Websites can sometimes overload the user.

The combination of Website complexity and low quality is potentially lethal to Company goals. Unhappy users will quickly depart for a different site; and, they probably won't leave with a good impression.

2.4.5 Website Architectural Factors

A Website can be quite complex, and that complexity -- which is what provides the power, of course -- can be a real impediment in assuring Website Quality. Features considered from Quality perspective:

Browser: The browser is the viewer of a Website and there are *so* many different browsers and browser options that a well-done Website is probably designed to look good on as many browsers as possible. This imposes a kind of *de facto* standard: the Website must use only those constructs that work with the *majority* of browsers. But this still leaves room for a lot of creativity, and a range of technical difficulties. And, multiple browsers' renderings and responses to a Website have to be checked.

Display Technologies: What you see in your browser is actually composed from many sources:

- *HTML.* There are various versions of HTML supported, and the Website ought to be built in a version of HTML that is compatible.
- *Java, JavaScript.* JavaScript and Java applets will be part of any serious Website, the quality process must be able to support these.
- *Database Access.* Interaction with database should be performed easily without any performance bottlenecks.

Navigation: Users move to and from pages, click on links, click on images (thumbnails), etc. Navigation in a Website is often complex and has to be quick and error free.

Object Mode: The display you see changes dynamically; the only constants are the "objects" that make up the display. These aren't real objects in the OO sense; but they have to be treated that way. So, the quality test tools have to be able to handle URL links, forms, tables, anchors, buttons of all types in an "object like" manner so that validations are independent of representation.

Server Response : Server response is a determining factor of the over all quality of a website.

Interaction & Feedback : For passive, content-only sites the only real quality issue is availability. For a Website that interacts with the user, the big factor is how fast and how reliable that interaction is.

Concurrent Users : Multiple user support should be there. While Websites often resemble client/server structures, with multiple users at multiple locations a Website can be much different, and much more complex, than complex applications.

2.4.6. Website Test Requirements

Test Context : Tests need to operate from the browser level for two reasons: (1) this is where users see a Website, so tests based in browser operation are the most realistic; and

(2) tests based in browsers can be run locally or across the Web equally well. Local execution is fine for quality control, but not for performance measurement work, where response time *including* Web-variable delays reflective of real-world usage is essential.

Website Dynamic Validation : Confirming validity of what is tested is the key to assuring Website quality.

Operational Testing : Individual test steps may involve a variety of checks on individual pages in the Website:

- *Page Consistency.* Consistency between pages is checked across different versions.
- *Table, Form Consistency.* To check the content of tables for consistency and accuracy.
- *Page Relationships.* Checking for the links consistency and to find missing and broken links.
- *Performance Consistency, Response Times.* Response time for a user action should be same and within the specified range.

Test Suites: Tests can be run in a variety of modes:

- *Unattended Testing.* Individual and/or groups of tests should be executable singly or in parallel from one or many workstations.
- *Background Testing.* Tests should be executable from multiple browsers running "in the background" on an appropriately equipped workstation.
- *Distributed Testing.* Independent parts of a test suite should be executable from separate workstations without conflict.
- *Performance Testing.* Timing in performance tests should be resolved to the millisecond; this gives a strong basis for averaging data.
- *Random Testing.* There should be a capability for randomizing certain parts of tests.
- *Error Recovery.* While browser failure due to user inputs is rare, test suites should have the capability of re synchronizing after an error.

Content Validation : Apart from how a Website responds dynamically, the content should be checkable either exactly or approximately. Here are some ways that content validation could be accomplished :

- *Structural.* All of the links and anchors should match with prior "baseline" data.
- *Checkpoints, Exact Reproduction.* One or more text elements -- or even all text elements -- in a page should be markable as "required to match".

2.5 Regression Testing

To validate the correctness and influence of the changes, the regression testing must be carried out. Regression testing is the process of testing modified software to detect whether new faults have been introduced into previously tested code. One approach to regression testing is to save the test suites for a product and reuse them to retest the product after it is modified, but this retest-all technique can be excessively expensive. Regression testing refers to testing the modified version of a system, **w'**, using a test set **T** used previously to test the original system, **w**. The selection of suitable test cases from **T** can be made in different ways and a number of regression-testing methods have been proposed. These methods are based on different objectives and techniques, such as:procedure and class firewalls , semantic differencing ; textual differencing slicing-based data-flow technique, test case reduction and safe algorithm based on program's control graph.

. Studies suggest that a significant portion of development and maintenance costs go to this retesting, which is known as *regression testing*. Reports estimate that regression testing consumes as much as 80% of the overall testing budget and can consume up to 50% of the cost of software maintenance. Rapidly changing software and computing environments present many challenges for effective and efficient regression testing in practice. Regression testing can be performed after changes are made to the software, such as after nightly or regular builds, before a new version of the software is released, every time the software is saved and compiled, such as in an agile development environment, or before patches, such as security patches, are released. Regardless of the environment or when it is performed, the goals of regression testing are the same: to improve confidence that the changes behave as intended and that they have not adversely affected unchanged parts of the software. Because regression testing is important, but expensive, much research has been performed, both in industry and in academia, to develop techniques to make regression testing more effective and efficient. This research has also produced many tools and systems that have been used for empirical studies that investigate the effectiveness, scalability, and practicality of the techniques. Researchers have developed techniques for addressing a number of issues related to regression testing. First, techniques attempt to *reduce* the regression testing time by creating effective regression test suites that test the changed part of the software, by identifying test cases in the regression test suite that do not need to be rerun on the changed software, and by identifying and removing obsolete test cases. Second, techniques can *reuse* test suites created for one version of the software by identifying those test cases that need to be rerun for testing subsequent versions of the software and by computing an effective ordering for running the test cases. Third, techniques can *recycle* test cases by monitoring executions to gather test inputs that can be used for retesting and by creating unit test cases from system test cases. Finally, techniques could *recover* test cases by identifying, manipulating, and transforming obsolete test cases, by generating new test cases from old ones, and by repairing test cases when new changes are introduced in the software for maintenance purposes.

2.5.1Regression Test Selection Techniques

As developers maintain a software system, they periodically *regressions test* it, hoping to find errors caused by their changes, and provide confidence that their modifications are correct. To support this process, developers often create an initial test suite, and then reuse it for regression testing. The simplest regression testing strategy, *retest all*, reruns every test case in the initial test suite. This approach, however, can be prohibitively expensive rerunning all test cases in the test suite may require an unacceptable amount of time. An alternative approach, *regression test selection*, reruns only a subset of the initial test suite. Of course, this approach is imperfect as well regression test selection techniques can have substantial costs, and can discard test cases that could reveal faults, possibly reducing fault detection effectiveness. These tradeoffs between the time required selecting and running test cases and the fault detection ability of the test cases that are run is central to regression test selection.

Minimization Techniques

Minimization-based regression test selection techniques hereafter referred to as *minimization techniques*, attempt to select minimal sets of test cases from original test set *T*, that yield coverage of modified or affected portions of program *P*.

Dataflow Techniques

Dataflow-coverage-based regression test selection techniques hereafter referred to as *dataflow techniques*, select test cases that exercise data interactions that have been affected by modifications.

.

Safe Techniques

Most regression test selection techniques | minimization and dataflow techniques among them | are not designed to be *safe*. Techniques that are not safe can fail to select a test case that would have revealed a fault in the modified program. In contrast, when an explicit set of safety conditions can be satisfied, safe regression test selection techniques guarantee that the selected subset, *T* , contains all test cases in the original test suite *T* that can reveal faults in *P*.

.

Ad Hoc / Random Techniques

When time constraints prohibit the use of a retest-all approach, but no test selection tool is available, developers often select test cases based on \hunches", or loose associations of test cases with functionality. One simple approach is to randomly select a predetermined number of test cases from *T*.

2.5.2 Regression Testing Process

The most crucial phase in the software development life cycle is maintenance phase, in which the development team is supposed to maintain the software which is delivered to the clients by them. Software maintenance results for the reasons like error corrections, enhancement of capabilities, deletion of capabilities and optimization. Now the changed or modified software needs testing known as regression testing. Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of obsolete capabilities. These modifications in the software may cause the software to work incorrectly and may also affect the other parts of the software, so to prevent this Regression testing is performed. Regression testing is used to *revalidate* the modifications of the software. Regression testing is an expensive process in which test suites are executed ensuring that no new errors have been introduced into previously tested code.

Let P be a program, let P′ be a modified version of P, and let T be a test suite for P. Regression testing consists of reusing T on P′, and determining where the new test cases are needed to effectively test code or functionality added to or changed in producing P′. There are various regression testing techniques (1) Retest all; (2) Regression Test Selection; (3) Test Case Prioritization; (4) Hybrid Approach. Figure 1 shows various regression testing techniques. Figure1. Regression Testing Techniques

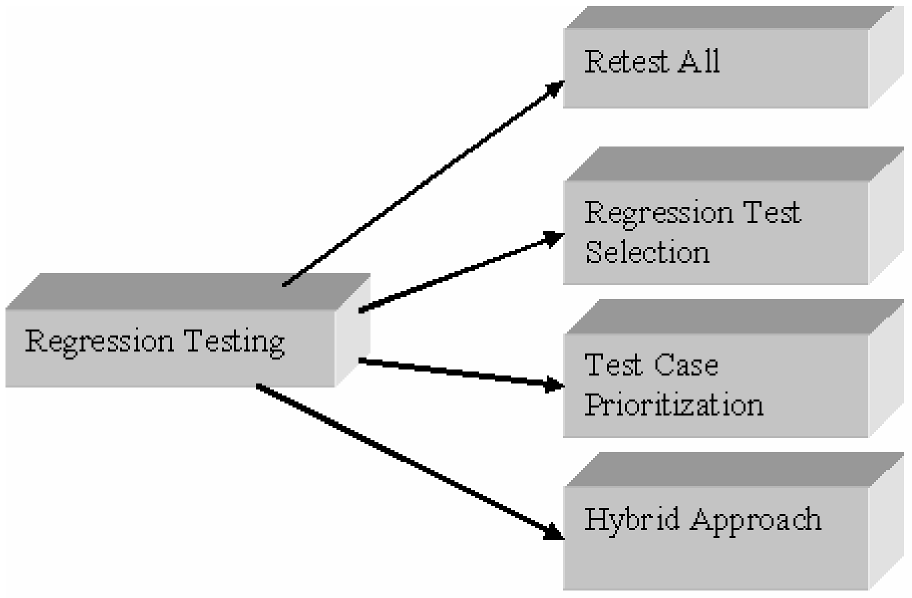


Figure 2.4: Regression Testing Techniques

(2) **Retest All**

Retest all method is one of the conventional methods for regression testing in which all the tests in the existing test suite are rerunned. So the retest all technique is very expensive as compared to techniques which will be discussed further as regression test suites are costly to execute in full as it require more time and budget.

(2) **Regression Test Selection (RTS)**

Due to expensive nature of "retest all" technique, Regression Test Selection is performed. In this technique instead of rerunning the whole test suite we select a part of test suite to rerun if the cost of selecting a part of test suite is less than the cost of running the tests that RTS allows us to omit. RTS divides the existing test suite into (1) Reusable test cases; (2) Retestable test cases; (3) 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.

1) **Coverage techniques:** they take the test coverage criteria into account. They find coverable program parts that have been modified and select test cases that work on these parts.

2) **Minimization techniques:** they are similar to coverage techniques except that they select minimum set of test cases.

3) **Safe techniques:** they do not focus on criteria of coverage, in contrast they select all those test cases that produce different output with a modified program as compared to its original version.

Various categories in which Regression Test Selection Technique can be evaluated and compared are: (a) Inclusiveness; (b) Precision; (c) Efficiency; (d) Generality.

a) **Inclusiveness** is the measure of extent up to which a technique chooses the test cases which will cause the changed program to produce different output than the original program, resulting in exposure of faults due to modifications.

b) **Precision** is the measure of ability of technique to prevent choosing test cases that will not make the changed program to produce different output than the original program.

c) **Efficiency** measures the practicality (computational cost) of a technique.

d) **Generality** is the measure of ability of a technique to handle complex modifications, realistic language constructs and realistic testing applications.

Various *techniques* of Regression Test Selection as given by various researchers are:

1) **<u>Modified non core function technique</u>:** defined in selects test cases that exercise functions in program that have been changed or deleted in producing changed program, or that exercise functions using variables or structures that have been deleted or changed in producing changed program.

2) **<u>Modification focused Minimization technique:</u>** seeks a subset of test suite that is minimal in covering all functions in program identified as changed.

3) **<u>Coverage focused Minimization technique</u>:** uses the suite reduction technique to find a subset of test suite that is minimal in covering all functions in program.

(3) **Test Case Prioritization**

This technique of regression testing prioritize the test cases so as to increase a test suite‘s rate of fault detection that is how quickly a test suite detects faults in the modified program to increase reliability. This is of two types:(1) General prioritization which attempts to select an order of the test case that will be effective on average subsequent versions of software .(2)Version Specific prioritization which is concerned with particular version of the software.

There are 18 different test case prioritizations techniques which are divided into three groups as shown in figure.

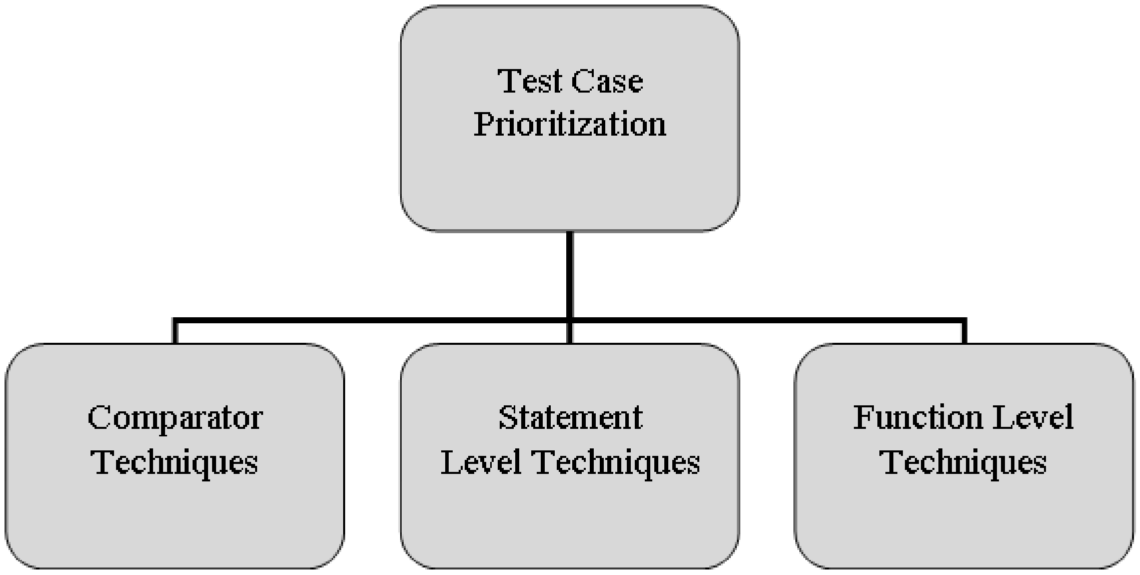


Figure 2.5: Classification of Test Case Prioritization

Comparator techniques:

Random ordering: in which the test cases in test suite are randomly prioritized. P2: Optimal ordering: in which the test cases are prioritized to optimize rate of fault detection. As faults are determined by respective test cases and we have programs with known faults, so test cases can be prioritized optimally.

Statement level techniques: (Fine Granularity)

Total statement coverage prioritization: in which test cases are prioritized in terms of total number of statements by sorting them in order of coverage achieved. If test cases are having same number of statements they can be ordered pseudo randomly.

Additional statement coverage prioritization: which is similar to total coverage prioritization, but depends upon feedback about coverage attained to focus on statements not yet covered. This technique greedily selects a test case that has the greatest statement coverage and then iterates until all statements are covered by at least one test case. The moment all statements are covered the remaining test cases undergo Additional statement coverage prioritization by resetting all statements to "not covered".

Function level techniques: (Coarse Granularity)

Total function coverage prioritization: it is similar to total statement coverage but instead of using statements it uses functions. As it has got coarse granularity so the process of collecting function level traces is cheaper than the process of collecting statement level traces in total statement coverage.

Additional function coverage prioritization: it is similar to Additional statement coverage prioritization with only difference that instead of statements, it is considering function level coverage.

(4) **Hybrid Approach**

The fourth regression technique is the Hybrid Approach of both Regression Test Selection and Test Case Prioritization. There are number of researchers working on this approach and they have proposed many algorithms for it. For example,

1) Hybrid technique proposed by Wong et al which combines minimization, modification and prioritization based selection using test history [21].

2) Hybrid technique proposed by Yogesh Singh et al is based on Regression Test Selection and Test Case Prioritization.

Regression testing is done in the maintenance phase of the software development life cycle to retest the software for the modifications it has undergone. Approximately 50% of the software cost is involved in the maintenance phase so researchers are working hard to come up with best results by developing new Regression Testing techniques so that the expenditure made in this phase can be reduced to some extent. Many foundations like US National Science Foundation (NSF), Galileo Research Group at Oregon State University, Mapstext Group at University of Nebraska Lincoln, Aristotle Research Group at Georgia Institute of Technology, NCSU Software Engineering Realsearch Group, National Natural Science Foundation of China are few of the many names who are working on development of new regression testing techniques and improving existing techniques on different aspects.The main aim is to fix the bugs and to make sure the problems are solved. Regression testing is acting as the control measure for maintaining the quality of the product with its specified requirements and the code[20].

2.6 Modelling web Application

Models represent a solid starting point for the implementation of a Web application taking into account static and dynamic aspects of the content, hypertext, and presentation levels of a Web application.While the content model of a Web application which aims at capturing underlying information and application logic is similar to the corresponding model of a non-Web application, the need to consider the hypertext is particular to Web applications. The hypertext model represents all kinds of navigation possibilities based on the content. The presentation model maps hypertext structures to pages and their links thus represent the graphical user interface. The inclusion of context information, such as user, time, location, and device used, and the adaptation of the Web application which is ‘‘derived’’ from this information, has gained increasing attention in modeling efforts. This is undoubtedly a consequence of ubiquitous Web applications that have become increasingly popular. This chapter discusses the spectrum of existing methods and some tools available to model Web applications and their highlights to help the reader select a suitable modeling method. Such methods are the basis for model-based development and code-generation tools, which allow us to consider the use of different Web clients and run-time platforms.

To model Web applications, the document-like character of its content as well as its non-linear hypertext navigation has to be taken into account. This is the reason why we distinguish three levels when modeling Web applications, as shown in Figure, in contrast to the two levels used in the modeling methods for traditional applications. The three levels are *content*, i.e., the information and application logics underneath the Web application, *hypertext*, i.e., the structuring

of the content into nodes and links between these nodes, and the *presentation*, i.e., the user interface or page layout. Most methods which are used to model Web applications follow this separation into three levels (Fraternali 1999).

A clear separation of these three levels allows reuse and helps to reduce complexity. For example, we could specify a number of different hypertext structures that will do justice to the specific requirements of different user groups and used devices for a given content. The aim of a content model is the explicit definition of the information structure. Comparable to a database schema in data modeling this eliminates redundancies. This means that the structure of the information will remain unchanged, even if the information itself changes frequently.

2.7 Chapter Summary

In this chapter we had elaborated the literature review or the research work done on the Testing web application using regression testing.

CHAPTER 3

Design and Architecture

In this chapter we will elaborate the two paradigm approaches used to design such systems and discusses that which approach we had used in our Modelling. This chapter also involves the discussion of the two models Event Dependency graph (EDG) and Event Test Tree (ETT) with their detailed design, working, and algorithm. Chapter finally ends with an example.

3.1 Web Application Modelling

Web services and their underlying systems grow over time and need to be retested whenever there is a change, to verify that the quality has not regressed. Testing requires time and effort. If we have modified only a small part of the system, it should be possible to reuse existing tests provided that the impact of the changes made can be isolated. There are two goals of regression testing – assuring system stability and establishing confidence in continued software quality. In recent years, Web applications are attracting more and more users with their important characters of universality, interchangeability and usability. Compared with the traditional software, Web applications have special properties, such as numerous users, heterogeneous and autonomous environments, and focused on information publication, index and retrieval.Traditional testing methods cannot meet the demands of Web applications and a unique suit of testing system is needed to fulfill the tasks of Web testing.

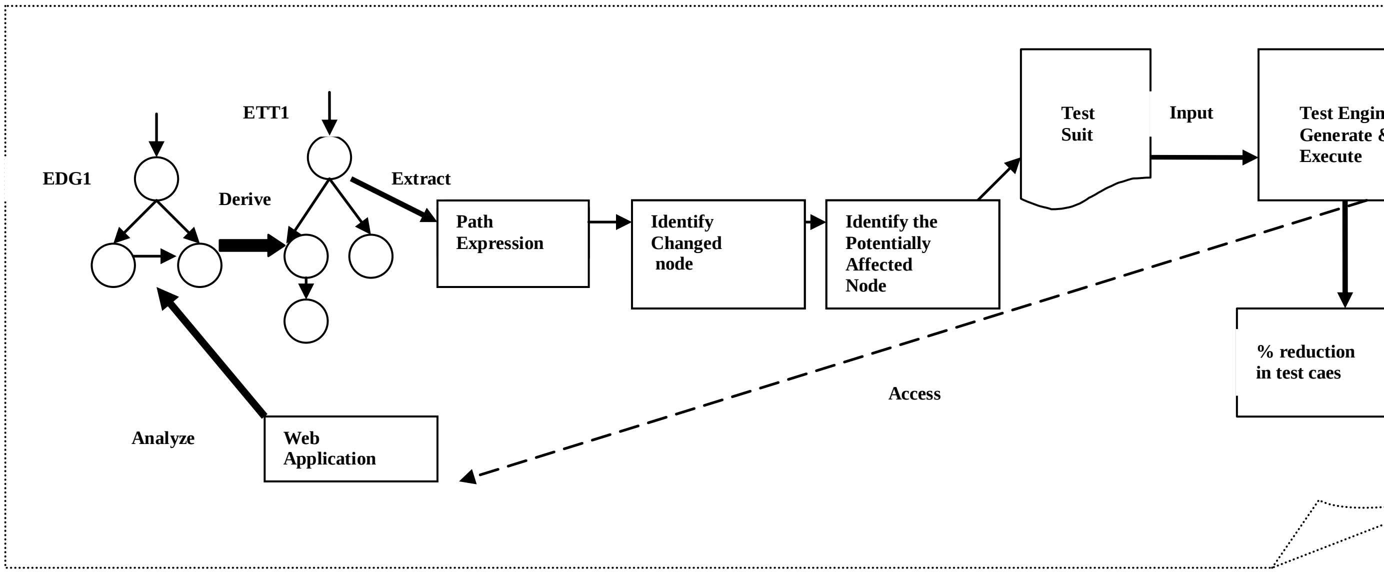


Figure 3.1 Architecture of System

3.1.1 Event Dependency Graph

.

Web applications can be viewed as event driven environment. Such applications are constructed by integrating interacting components that are invoked via events. The interaction is based on data dependence, control dependence and call dependence. The data dependence holds between two statements if one defines the value of a variable used by the other. The control dependence exists between elements of the same page if there is a flow control between the first element and the second element. The call dependence holds between a calling statement and a server program or procedure to be invoked. Such dependences are differentiated through two classification types, the internal and the inheritance call dependences.

Event-based dependence can be classified into three types which are the link, visible effect, and invisible effect dependences. The link dependence holds between two pages if the first requests the second via an event (usually by enabling a graphics element or a hyperlink). Visible effect dependence exists between two pages if the requesting page modifies the other via and event and the second page opens with the modified data. Invisible effect dependence holds between two pages when a page modifies another without displaying the effect. Semantic dependences hold between an informative object (graphics, textual, processing) and a page or another informative object if the former provides information to the latter.

To model web applications, we suggest an event dependency graph (EDG) that combines all of these dependences with interacting components. As for the link dependence, the requested page is represented in the EDG by a solid square arrow. The visible effect dependence is represented by a square dashed arrow points to the affected page.. The invisible effect dependence is represented by a square dotted arrow pointing to the affected page.

In order to model semantic dependence new notational elements to the dependency graph are introduced: web page (the building block of a web application), textual element (text included on a page), graphic (button,text-field, drop down, etc.), and event.p5 to p1 shows invisible effect with arrow pointing to affected page p1.p1 to p2 is link dependence as p1 is requesting p3 through a hyperlink.p5 to p6 is direct dependence as p5 opens p6 with some modified data.

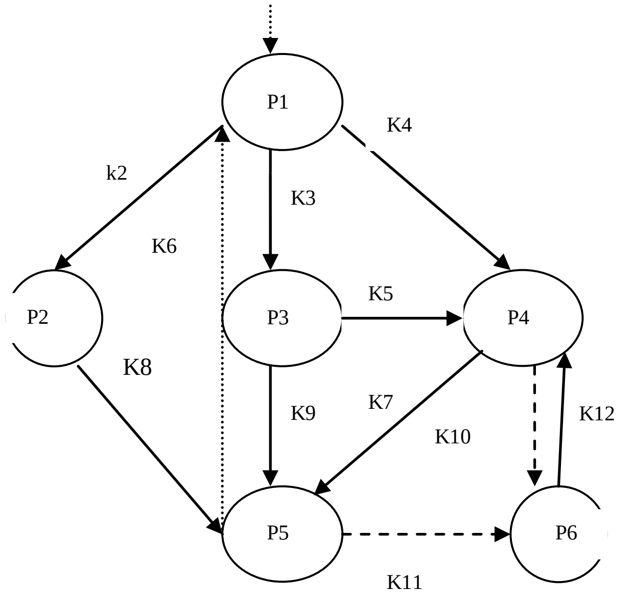


Figure 3.2 Event Dependency graph

3.1.2 Event Test Tree

With Event dependency graph the main problem is that we often cannot verdict the termination of a path and the path is cyclic, this will complicate the testing process, thus, driving the testing process using the EDG is difficult and unwieldy. Therefore, A ETT (Event Test Tree), which is based on the EDG is used. From the ETT, shorter paths can be generated than the EDG without the loss of page and link coverage. An ETT is a spanning tree constructed from an EDG,

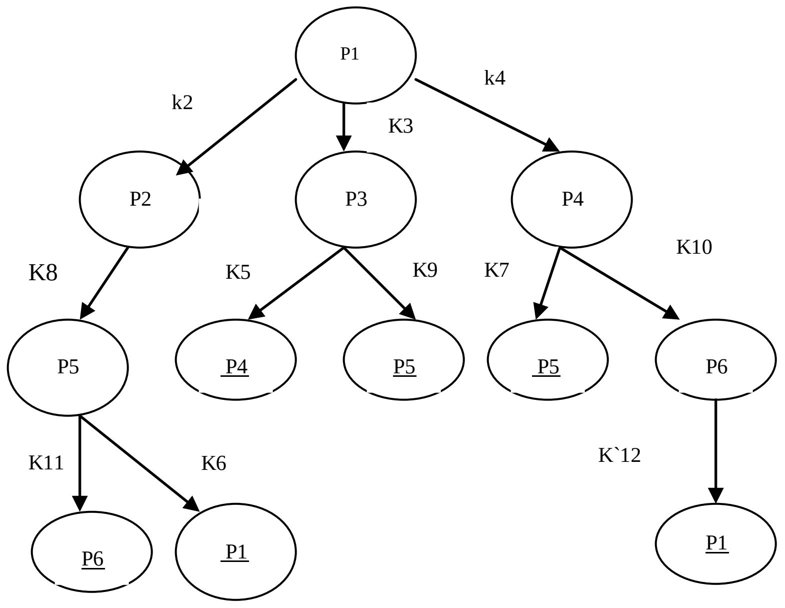


Figure 3.2   Event Test Tree

By coverting the EDG to ETT we obtain an expression and we get information that P1 is root node and p2, p3 and p4 are the child node of that. we also know that there is a path expression which shows path like P1->p2->p5(p6+p1).in next step we will identify the nodes which are changed and potentially affected nodes.

3.2 Algorithm

3.2.1 EDG to ETT

In step 1 we have made Event dependency graph which is followed by the link, visible effect, and invisible effect dependences. The link dependence holds between two pages if the first requests the second via an event (usually by enabling a graphics element or a hyperlink). Visible effect dependence exists between two pages if the requesting page modifies the other via and event and the second page opens with the modified data. Invisible effect dependence holds between two pages when a page modifies another without displaying the effect. Semantic dependences hold between an informative object (graphics, textual, processing) and a page or another informative object if the former provides information to the latter.

STEP	FIRST	SECOND
1	P1	NULL
2	P2,p3,p4	P1
3	P3,p4,p5	P1,p2
4	P4,p5,p4,p5	P1,p2,p3
5	P5,p4,p5,p5,p6	P1,p2,p3,p4
6	P4,p5,p5,p6,p6,p1	P1,p2,p3,p4,p5
7	p5,p5,p6,p6,p1	P1,p2,p3,p4,p5
8	p5,p6,p6,p1	P1,p2,p3,p4,p5
9	p6,p6,p1	P1,p2,p3,p4,p5
10	p6,p1	P1,p2,p3,p4,p5,p6
11	p6,p1	P1,p2,p3,p4,p5,p6
12	p1	P1,p2,p3,p4,p5,p6
13	Null	P1,p2,p3,p4,p5,p6

Table 3.1:EDG to ETT

EDG to ETT

```
Input: a EDG
Output: a ETT derived from the EDG
begin
(1) add the initial page identifier of the PFD into FIRST;
(2) if FIRST is empty, then go to (6);
(3) select the first page identifier denoted by pid from FIRST. If pid is within SECOND,
then go to (5).
        Otherwise, add it into the end of SECOND;
(4) if pid is linking to other pages, then
```

 if some of the other page identifiers are Within FIRST or SECOND, then generate their copies; retain the links between *pid* and the other pages (or their copies) of the PFD, and

 add the other page identifiers (or their copies) into the end of FIRST;

(5) delete *pid* from FIRST and then go to (2);

(6) output the derived PTT, which is the PFD with only the retained links.

End

The initial page (node) is called root page (node), the link entering into the root page is called root link, the leaf page (node) is called tail page, the link entering into the tail page is called tail link, and each of all the other pages (links) is called branch page (link).A tail page may also be a branch page or a root page.

Each page underlined is a copy of its corresponding page derived in one or more previous steps. Null denotes that FIRST or SECOND is empty. The pages in FIRST and SECOND, as shown in table 1, are changed during the process of generation using algorithm PFD2PTT.

3.2.2 Compare potentially affected and affected nodes.

In previous two sections we have made Event Dependency Graph and Event Test tree of Web Application we have made graph as well as tree for both original and modified web application. Now we compare both Event dependency tree to find affected nodes means nodes which are changed and potentially affected nodes.

The Event dependency graph of original web application and modified web application made in step 1 and then we have converted Event dependency graph in to Event test tree in step2 .Now we compare both ETT which represent the changed elements and dependences.

1. For each element pi in original ETT find the corresponding node pi’ in the modified ETT.

2. If pi’ is found in modified web application then for each neighbour in the original graph, find the corresponding neighbour in the modified graph

3. If we did not found element in the modified EDG means node has been removed from the modified web application. We consider this node as changed node.

In the Next step we find those nodes that may be affected by the change. The potentially affected nodes are the nodes that are connected either directly or indirectly to the affected nodes. All the dependences are considered to identify the affected nodes. For each neighbour of the affected node, add it to the affected nodes list and check its neighbours and do the same for the neighbours. The details of how the potentially affected nodes are selected are as follows:

1. Go through all the neighbours of the selected node
2 checks if the neighbour is already added to the selected nodes list do nothing
3. If the neighbour is not added to the list of selected nodes, add it to the list. Each new Neighbour is considered as the selected node, repeat the procedure from (1) for each new Neighbour.

Second Method is

1. Select a node that has still not been selected from the original tree..
2. Find all the child nodes of the selected node.
3. Store the nodes in 2 d array with first element as the selected node and rest elements as the children of selected node,
4. Repeat steps 1 to 4 till all the nodes have been covered.
5. Make a similar 2d array for the new tree.
6. Compare the 2 d array element by element.
7. If there is a difference in the two arrays in any row return the first element of the row i.e. the parent node.

   Using these Algorithms we can find affected and potentially affected nodes.now we got an text expression from tree. We give this expression to c compiler and test cases will be generated on basis of this.Finally we calculate reduction in test cases from original and modified one.

3.3 Test Case Reduction

Finally we will calculate reduction in Number of test cases.we find Test cases in original system and modified system and finally calculate the reduction in test cases

Test cases in original system : **a**

Test cases in modified system : **b**

No of test cases eliminated : **a-b**

Percentage reduction in test cases:

(a-b/a) X 100% = ?%

3.4 Chapter Summary

In this chapter we had elaborated the two algorithms and discuss the approach used in our model.This chapter also include the method how we have reduces the test cases..

CHAPTER 4

IMPLEMENTATION

In this chapter we will implement the design proposed in the previous chapter using some algorithms and coding.

Regression testing is an important part of maintenance. It is performed when the software is modified to ensure that it has not lost the desired functionality. Regression test suites can be expensive to execute in full; thus test engineers may adopt a suitable strategy to reduce the testing time. Many techniques are proposed for minimizing cost related to regression testing

While doing implementation we get to know the ultimate requirement of the users who is directly related to the system. The ways on which we are working now is quite tough and require a heavy type of requirement to satisfy the need of the researchers. Today is the world of challenge and at every moment of time we face a new kind of situation. To be at the top the system designed should be a best system and also the cost is one of the major factors. Today we have seen that people can switch over from one system to another system design and that is not because of bad performance but due to some other factors also. Some of the factors are fast response, accuracy etc. All these things are possible but to maintain now a days the cost is become a very crucial factor. Up to a certain level cost is immaterial but beyond that one cost matters.

To overcome the above problems and to provide a new intelligent system one has to move from the current environment to another one or modify the existing environment. Now before going to implement the system we can recollect the main things that are the part of the thesis.

In this we have developed a website Student Information System. Which contains all the student Information. Student can login, download study material, can check their fee status etc.

Next we have made Event dependency graph of original and modified web application. In Next Step we have converted Event dependency graph to Event Test Tree. Now we got a Test expression which is known as test path. Now we have given this to c compiler and .finally generated test cases of original and modified system and calculates the reduction in test cases.

In section 4.1 we have given all the details and techniques we have used to implement our website student Information system.

In section 4.2 we have made Event Dependency Graph and Event test Tree of original and modified web application. Now we compare original and modified web application and compare the affected and potentially affected nodes.

In section 4.3 we have given this test expression to c compiler and In section 4.4 we calculate the reduction in number of test cases

4.1 Technology Used

We have developed a web application Student Information system in J2EEE. The Java 2 Platform, Enterprise Edition (J2EE) defines the standard for developing multitier enterprise applications. The J2EE platform simplifies enterprise applications by basing them on standardized, modular components, by providing a complete set of services to those components, and by handling many details of application behaviour automatically without complex programming. There are four tiers in J2EE application model: client tier, web tier, business tier and enterprise information system (EIS) tier. Business and web tiers are generally existed in a server application called application server or J2EE server. Application server provides complex services needed by components in business and web tiers.

4.1.1. HARDWARE & SOFTWARE SPECIFICATION:

- **Technology Used :** J2EE(Advanced Java)

- **Operating System Used:** Windows XP

- **Hardware Used:** Computer with a broadband Internet connection

- **Software Used:** Net Beans IDE 6.8

- **Backend Used:** MS Access

- **Server Used:** Apache Tomcat and Glassfish

4.2 Methodology

In our proposed system, we have made Event dependency graph of our original and modified web application. **Event Dependency Graph** (EDG) combines all of these dependences with interacting components. In link dependence, the requested page is represented in the EDG by a solid square arrow. The link dependence holds between two pages if the first requests the second via an event (usually by enabling a graphics element or a hyperlink). Visible effect dependence exists between two pages if the requesting page modifies the other via and event and the second page opens with the modified data.. The visible effect dependence is represented by a square dashed arrow points to the affected page.The invisible effect dependence is represented by a square dotted arrow pointing to the affected page. Invisible effect dependence holds between two pages when a page modifies another without displaying the effect.

4.2.1 Event Dependency Graph of original Application

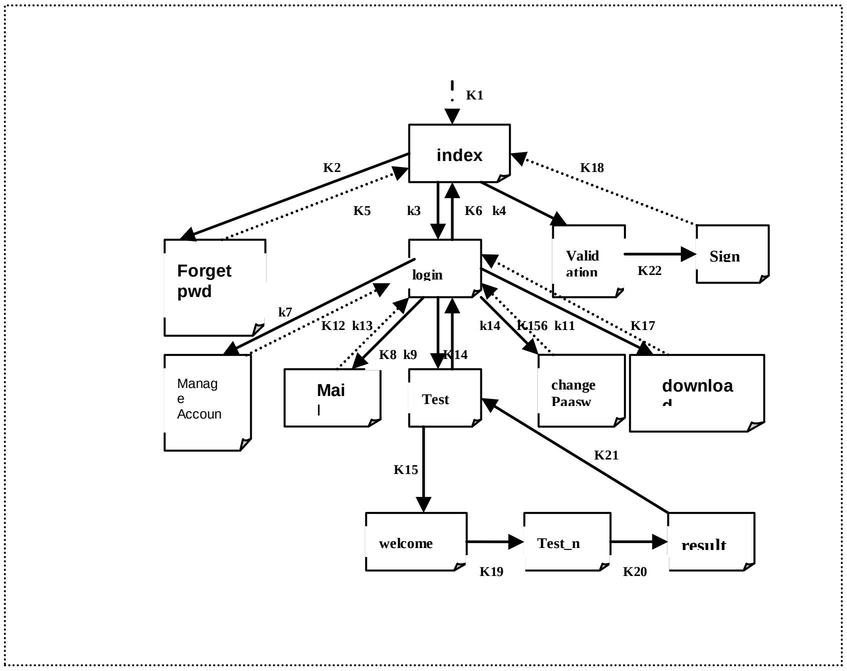
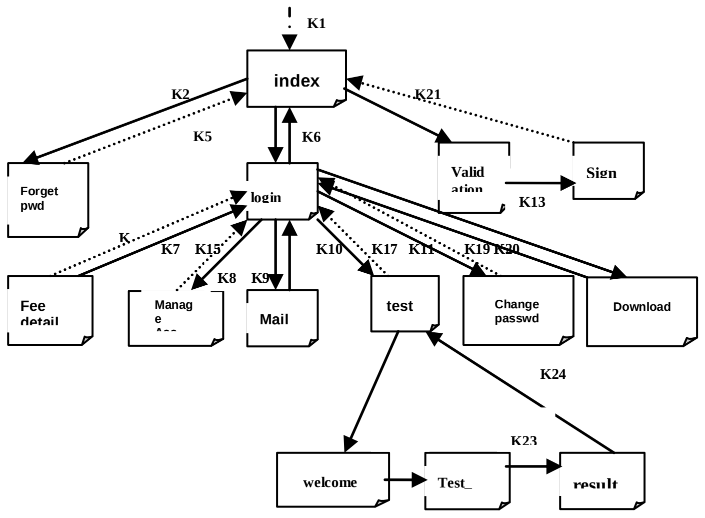


Figure 4.1 Event Dependency Graph of original Application

KI is default link. Index page is first page to which student can log in. he can go to forget password link,can login,can sign in and can do validation as shown in Figure. After login he can manage his account can mail,test,download as well as can change his password.In test in web page he can give test and can see the result.

4.2.2 Event Dependency Graph of modified Web Application

In modified web application we have added one fee detail page if student have paid fees then only he can log in and can give test else not.other links are same like change password,manage account etc.



4.2 Event Dependency Graph of Modified Application

4.2.3 Event Test Tree of original Web Application

The cyclic redundancy is avoided in this in which we have converted graph of original and modified web application to tree.

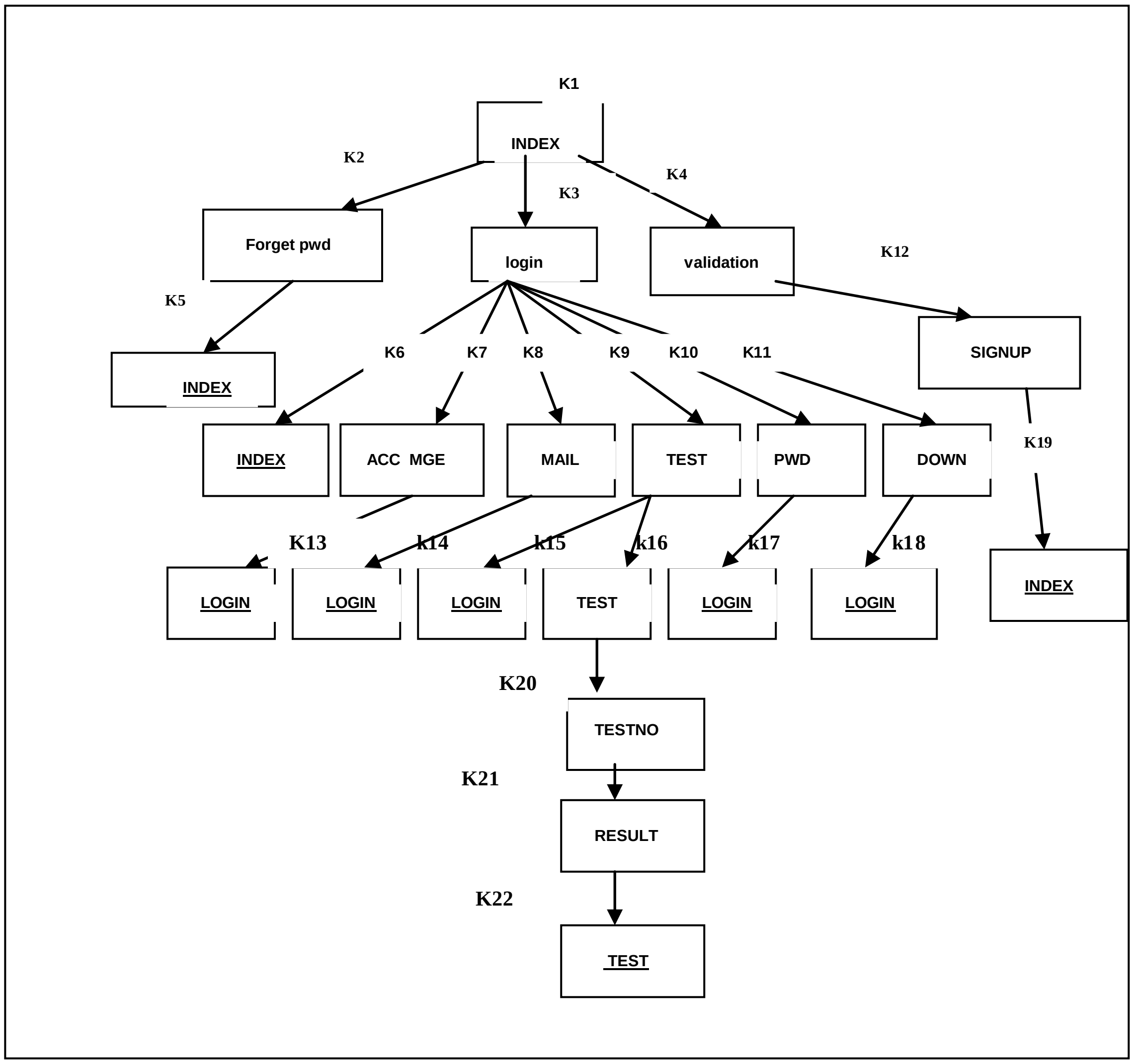


Figure 4.3 Event Test Tree of original application

4.2.4 Event Test Tree of Modified Web Application

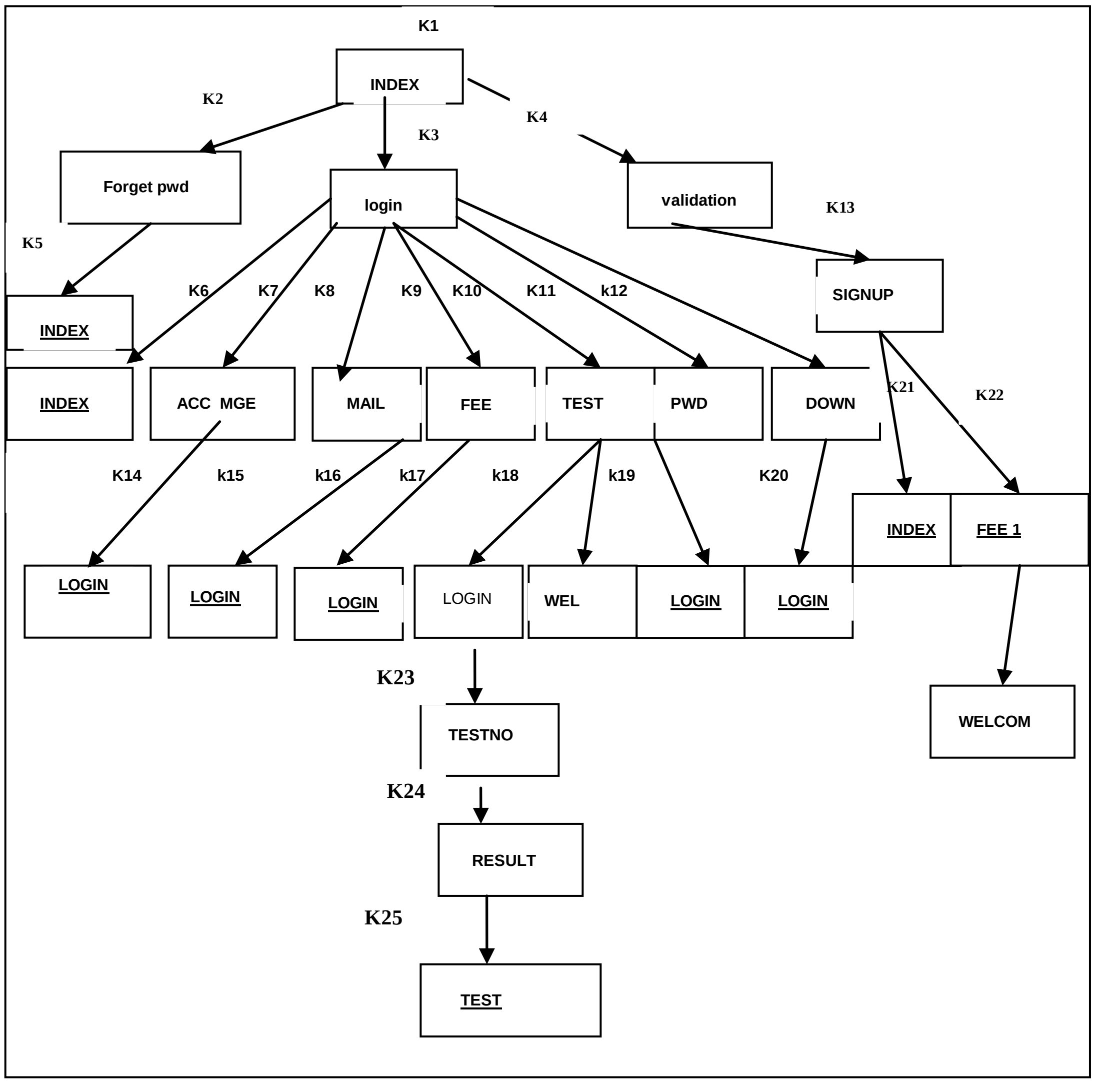


Figure 4.4 Event Test Tree of modified Application

4.2.5 Test Path

FOR ORIGINAL SYSTEM

Path expression
k1→(k2→k5+k3→(k6+k7→k13+k8→k14+k9→(k15+k16→k20→k21→k22)+k10→k17+k11→k18)+k4→k12→k19

Test path
k1→k2→k5
k1→k3→k6
k1→k3→k7→k13
k1→k3→k8→k14
k1→k3→k9→k15
k1→k3→k9→k16→k20→.k21→k22
k1→k3→k10→k17
k1→k3→k11→k18
k1→k4→k12→k19

FOR MODIFIED SYSTEM

Path expression
k1→(k2→k5+k3→(k6+k7→k14+k8→k15+k9→k16+k10→(k17+k18→k23→k25 →k26)+k11→k19+k12→k20)+k4→k13→(k21+k22→k24)

Test path
k1→k2→k5
k1→k3→k6
k1→k3→k7→k14
k1→k3→k8→k15
k1→k3→k9→k16
k1→k3→k10→k17
k1→k3→k10→k18→k23→.k25→k26
k1→k3→k11→k19
k1→k3→k12→k20
k1→k4→k13→k21
k1→k4→13→k22→k24

4.3 Analysis and Results

We had implemented the above mentioned Test Expression and generated Test cases as shown below.

4.3.1 Test Cases for original System

Test ID	**Description**	**Input**	**Expected result**	**Actual Result**
General				
1 .	Entering URL in to the web browser and clicking GO button	URL	Site opens	Site opens
2 .	Opening the website – checking for focus	-	Focus on user name text box	Focus on user name text box
3 .	TAB key control	-	Follow TAB key functions	Follows TAB key functions
4.	Checking proper working of links	-	All linked pages open	Links working properly
For Sign In				
5.	Entering valid user name and password on index page	Username and password	Respective login page opens	Respective login page opens
6 .	Entering invalid username and valid password	Username and password	Error message comes	Error message comes
7 .	Entering valid username and invalid password	Username and password	Error message comes	Error message comes
8 .	Entering valid username and leaving password field blank	Username	Error message comes	Error message comes
9 .	Entering invalid username and leaving password field blank	Username	Error message comes	Error message comes
10.	Entering valid password and leaving username field blank	Password	Error message comes	Error message comes
11.	Entering invalid password and leaving username field blank	Password	Error message comes	Error message comes

12.	Keeping both password and username field blanks	-	Error message comes	Error message comes
13.	Entering valid username and password in CAPS	Username and password	Error message comes	Login page comes
For Forget Password				
14.	Entering valid username, registration ID, security question and security answer	Username, registration ID, security question and security answer	Password comes	Password comes
15.	Entering invalid registration ID and valid username, security question and security answer	Username, registration ID, security question and security answer	Error message comes	Error message comes
16.	Entering valid username, security question and registration ID and invalid security answer	Username, registration ID, security question and security answer	Error message comes	Error message comes
17.	Entering invalid security question and valid registration ID, security answer and username	Username, registration ID, security question and security answer	Error messages comes	Error message comes
18.	Leaving registration ID field blank and entering valid username, security question and security answer	Username, security question and security answer	Error messages comes	Error message comes
19.	Leaving security answer field blank and entering valid username, security question and registration ID	Username, registration ID and security question	Error message comes	Error message comes
20.	Leaving username	Registration	Error	Error

	field blank and entering valid security question, security answer and registration ID	ID, security question and security answer	message comes	message comes
21.	Entering valid username in CAPS, security question, security answer and registration ID	Username, registration ID, security question and security answer	Error message comes	Password comes
For Sign Up				
22.	Entering invalid ID, name and type combination	ID, name and type	Account cannot be created message comes	Account cannot be created message comes
23.	Entering an already registered persons information	ID, name and type	You are already registered message comes	You are already registered message comes
24.	Entering valid name and type and leaving registration ID blank	Name and type	Account cannot be created message comes	Account cannot be created message comes
25.	Entering valid name, ID and type for a person registering for first time	Name, ID and type	Form 2 opens	Form 2 opens
For Sign Up (continued)				
26.	Leaving some field blank and clicking create button	-	Error message comes	Error message comes
27.	Not checking username availability	-	Not allowed to proceed further	Not allowed to proceed further
28.	Entering previously used username	-	Username not available message comes	Username not available message comes

29.	Entering password of less than 3 characters	-	Error message comes	Error message comes
30.	Confirm password and password field does not match	-	Error message comes	Error message comes
31.	Entering all valid and correct information	-	Your account is created message comes	Your account is created message comes
For Account Management				
32.	Entering all valid and correct information	-	Account is created message comes	Account is created message comes
33.	Leaving all the field blank and clicking create button	-	Error message comes	Error message comes
34.	Entering some information in valid and some in invalid format	-	Error message comes	Error message comes
35.	Entering some information in valid format and leaving some fields blank	-	Error message comes	Error message comes

Table 4.1 Test cases for Original System

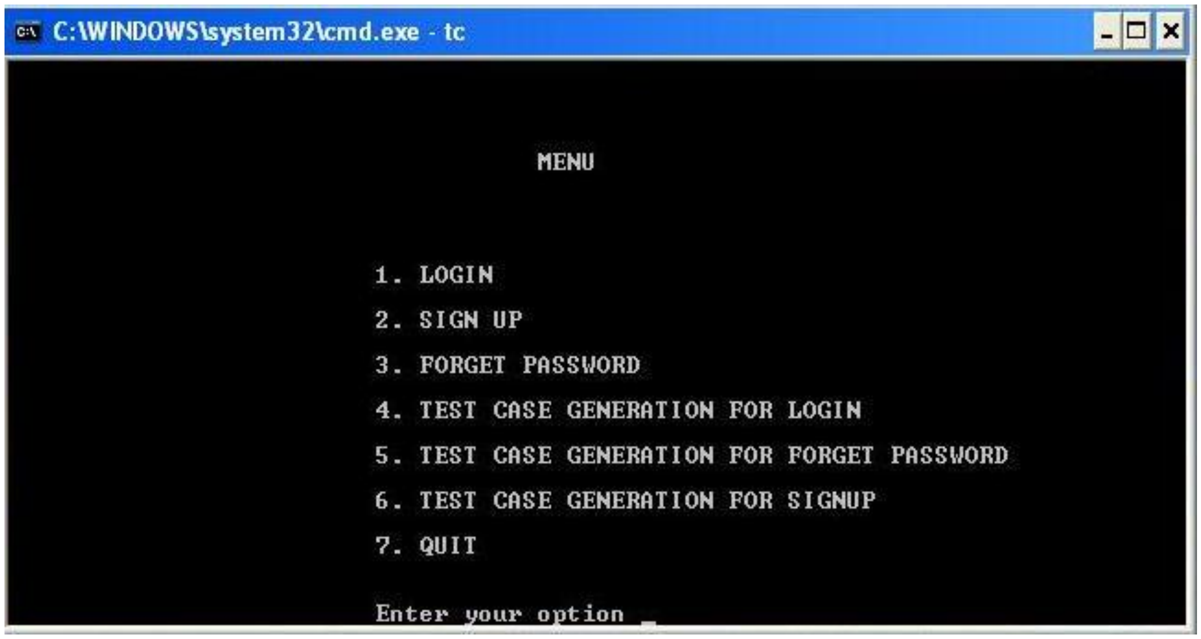
4.3.2. Test Cases for Modified System

Test ID	Description	Input	Expected result	Actual Result
General				
1.	Entering URL in to the web browser and clicking GO button	URL	Site opens	Site opens
For Sign In				
2.	Entering valid user name and password on index page	Username and password	Respective login page opens	Respective login page opens

3.	Entering invalid username and valid password	Username and password	Error message comes	Error message comes
4.	Entering valid username and invalid password	Username and password	Error message comes	Error message comes
5.	Entering valid username and leaving password field blank	Username	Error message comes	Error message comes
6.	Entering invalid username and leaving password field blank	Username	Error message comes	Error message comes
7.	Entering valid password and leaving username field blank	Password	Error message comes	Error message comes
8.	Entering invalid password and leaving username field blank	Password	Error message comes	Error message comes
9.	Keeping both password and username field blanks	-	Error message comes	Error message comes
10.	Entering valid username and password in CAPS	Username and password	Error message comes	Login page comes
For Sign Up				
11.	Entering invalid ID, name and type combination	ID, name and type	Account cannot be created message comes	Account cannot be created message comes
12.	Entering an already registered persons information	ID, name and type	You are already registered message comes	You are already registered message comes
13.	Entering valid name and type and leaving registration ID blank	Name and type	Account cannot be created message comes	Account cannot be created message comes
14.	Entering valid name, ID and type for a person registering for first time	Name, ID and type	Form 2 opens	Form 2 opens
For Sign Up (continued)				
15.	Leaving some field blank and clicking create button	-	Error message comes	Error message comes
16.	Not checking username availability	-	Not allowed to proceed further	Not allowed to proceed further
17.	Entering previously used username	-	Username not available message	Username not available

			comes	message comes
18.	Entering password of less than 3 characters	-	Error message comes	Error message comes
19.	Confirm password and password field does not match	-	Error message comes	Error message comes
20.	Entering all valid and correct information	-	Your account is created message comes	Your account is created message comes
For Account Management				
21.	Entering all valid and correct information	-	Account is created message comes	Account is created message comes
22.	Leaving all the field blank and clicking create button	-	Error message comes	Error message comes
23.	Entering some information in valid and some in invalid format	-	Error message comes	Error message comes
24.	Entering some information in valid format and leaving some fields blank	-	Error message comes	Error message comes

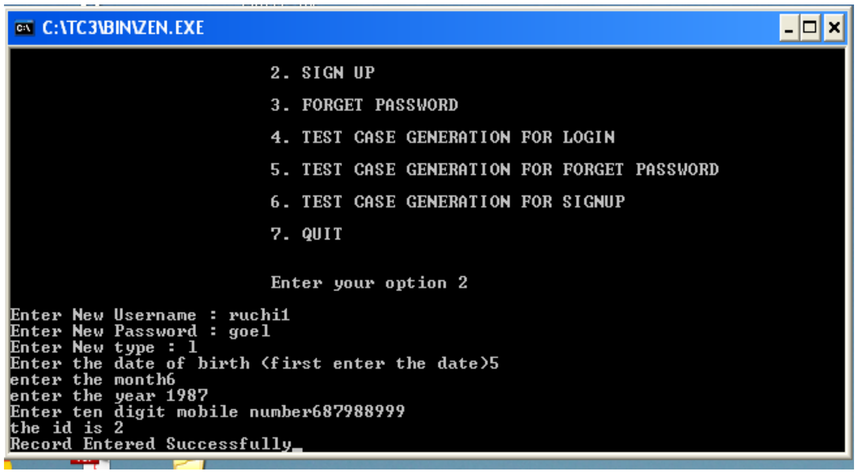
Table 4.2 Test cases for Modified System



```
                    MENU


          1. LOGIN
          2. SIGN UP
          3. FORGET PASSWORD
          4. TEST CASE GENERATION FOR LOGIN
          5. TEST CASE GENERATION FOR FORGET PASSWORD
          6. TEST CASE GENERATION FOR SIGNUP
          7. QUIT

          Enter your option _
```

Figure 4.5 Main Menu

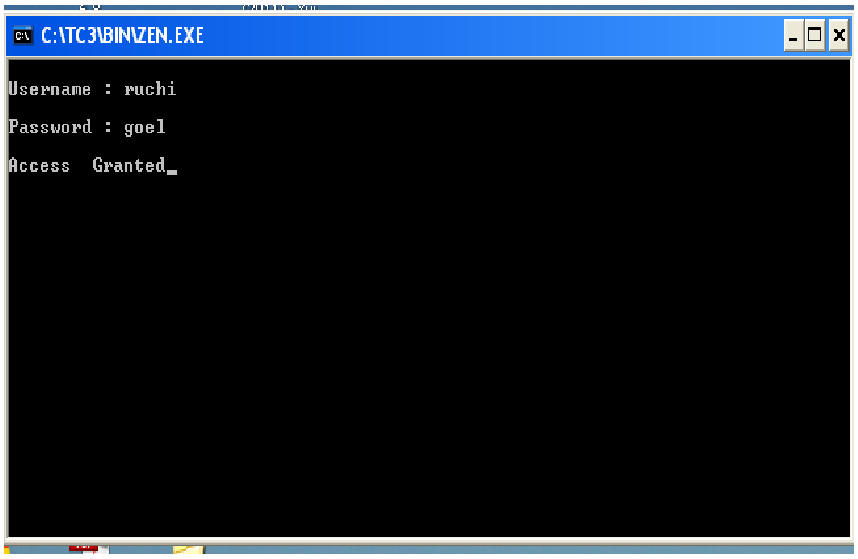
We have generated test cases using c.First we have to sign up as a new user.All records are maintained in signup format.



```
                                 2. SIGN UP
                                 3. FORGET PASSWORD
                                 4. TEST CASE GENERATION FOR LOGIN
                                 5. TEST CASE GENERATION FOR FORGET PASSWORD
                                 6. TEST CASE GENERATION FOR SIGNUP
                                 7. QUIT

                                 Enter your option 2
Enter New Username : ruchi1
Enter New Password : goel
Enter New type : 1
Enter the date of birth (first enter the date)5
enter the month6
enter the year 1987
Enter ten digit mobile number687988999
the id is 2
Record Entered Successfully_
```

4.6 Signup Menu

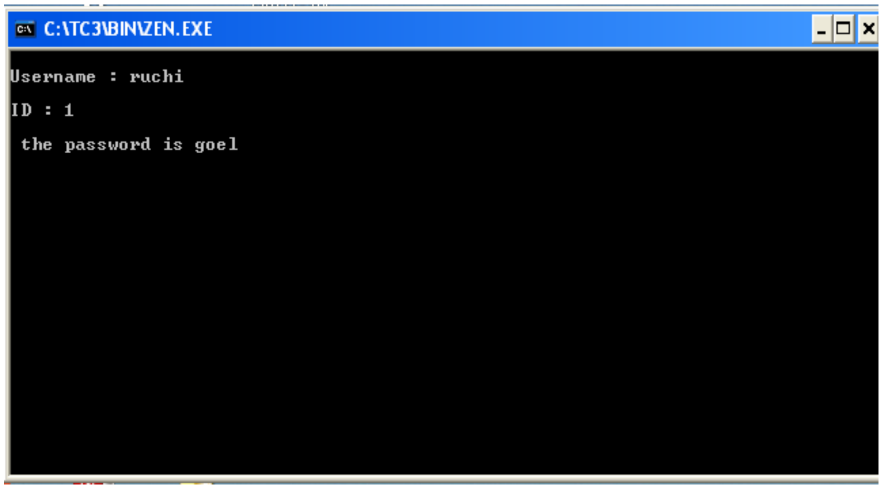
After signing up user can login by using their user name and password.If username and password matched than access granted else denied.



```
Username : ruchi

Password : goel

Access  Granted_
```

4.7 login Entry

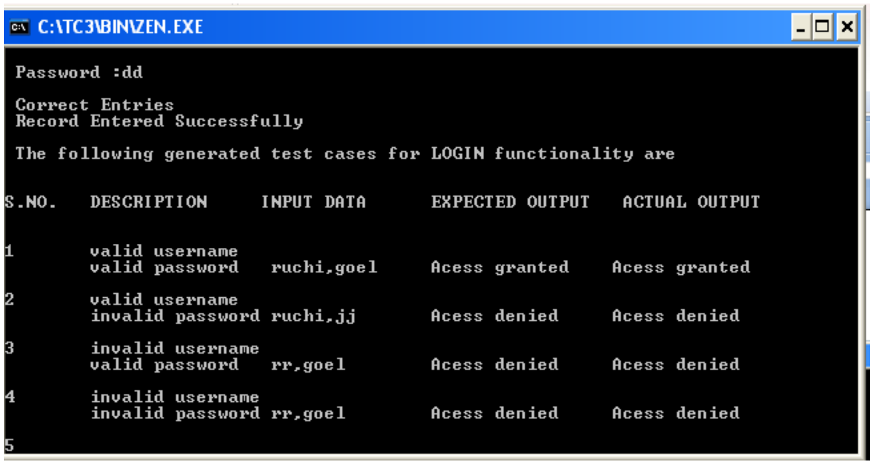
If user forgets his password he can retrieve his password by giving his id and user name.



```
Username : ruchi
ID : 1
 the password is goel
```

4.8 Forget Password Menu

Suitable test cases can be generated for login functionality by giving valid and invalid entries.expected and actual results are compared



```
 Password :dd

 Correct Entries
 Record Entered Successfully

 The following generated test cases for LOGIN functionality are

S.NO.   DESCRIPTION     INPUT DATA      EXPECTED OUTPUT   ACTUAL OUTPUT

1       valid username
        valid password   ruchi,goel     Acess granted    Acess granted

2       valid username
        invalid password ruchi,jj       Acess denied     Acess denied

3       invalid username
        valid password   rr,goel        Acess denied     Acess denied

4       invalid username
        invalid password rr,goel        Acess denied     Acess denied

5
```

4.9 Test cases for LogIn Functionality

Testcases of forgetpassword and signup are attached in appendix

4.4 Result

Test cases in original system: **35**

Test cases in modified system: **20**

No of test cases eliminated: **15**

Percentage reduction in test cases:

(15/35) X 100% = 42.8%

4.5 Chapter Summary

In this chapter we had elaborated the implementation of the both original and modified system. We had also discussed all the test cases used in the system and how these test cases are used in the system at different . We had also discussed the analysis and result of the system and shows screenshots of testcases generated.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1 Conclusion:

Regression testing is selective retesting of the system; executed with an objective to ensure the bug fixes work and those bug fixes have not caused any un-intended effects in the system. A "final regression testing" is done to validate the gold master builds, and "Regression testing" is done to validate the product and failed test cases between system test cycles.

The final regression test cycle is conducted on an "unchanged build for a period of x days" or for a period, which was agreed as the "cook-time" for release. The product is continuously exercised for the complete duration of this cook-time. Some of the test cases are even repeated to find out whether there are failures in the final product that will reach the customer. All the bug fixes for the release should have been completed for the build used for the final regression test cycle. The final regression test cycle is more critical than any other type or phase of testing, as this is the only testing which ensures "the same build of the product that was tested reaches the customer".

A normal regression testing can use the builds for a time period that is needed for the test cases to be executed. However unchanged build is highly recommended for each cycle of regression testing.

A web application entitled “Student Information System” was developed using J2EE technology. For the system test cases were generated and the Event Dependency Graph (EDG) was drawn. Then the system was modified by adding the FEE DETAIL module and EDG was drawn for the modified system. With the help of the EDG. An algorithm was then employed to convert EDG to Event Test Tree’s (ETT’s). ETT’s of both, the original and the modified system were compared to get the changed and the potentially affected nodes and on this basis test case subset was selected from the original test case set. The test case reduction percentage was calculated and was compared with the one obtained from EDG comparison technique to obtain the most effective method for test case reduction.

A new technique to select test cases for regression testing of web applications based on the Edge Test Tree is implemented. The regression test selection technique is based on identifying changed and potentially changed components. Empirical results show that this technique selects a reduced number of test cases and in our implementation of the same, 42% of the tests were eliminated, yet the selected test sequences cover the whole of the modified system.

Then we compared the test case reduction percentage of EDG methodology and ETT technique and came to the conclusion that ETT is a more effective method for the test cased reduction.

Finally, the above results show that the proposed ETT technique leads to greater reduction in the number of test cases to be rerun for the modified system testing. However, the reduced set of test cases still covers affected and potentially affected components that are determined by the program change.

The main Problem with the proposed system:

First, Tree is not an AVL balanced tree and it is not possible to create it for our web application because our home page is first page to which hyperlink to all pages are there . and with AVL tree our first page will not be home page and nodes in left are less than nodes in right.

Second, test expression must be calculated using some proposed algorithm.

Third, this work is more theoretical and has less practical applications.

5.2 Future Work:

The model proposed and designed in this thesis is only being executed on the predefined and assumed test cases. Automatic test cases can be generated by this method.

The proposed methodology for testing in this project helps reduce the test cases to a very large extent. There are various other methods besides the Edge Test Tree on which we can work upon, for example Ricca, a scientist proposed a UML model of web application and propose that all paths that satisfy selected criteria be selected for regression testing.

We can also deploy **System Dependent Graph (SDG)** instead of (EDG). It is based on slicing. But the disadvantage of using this methodology is that it can increase the workload and cost of the testing process.

Various other techniques have been mentioned in the paper which can be implemented and the results thus obtained can be compared with the present findings

We have made Event Dependency graph and converted that graph to Event Test Tree.A suitable algorithm is provided.we can make that automatically
Means some automation can be done to convert that graph to tree.

BIBLIOGRAPHY

Bibliography

1. Yogesh Singh,Arvinder Kaur,Bharti SuriA New Technique for Version – Specific Test Case Selection and Prioritization for Regression Testing, Journal of the CSI

2. Praveen Ranjan Srivastava,Test case priortization Journal of Theoretical and Applied Information Technology

3. Tarhini, Z. Ismail and N. Mansour. Regression Testing Web Applications. International Conference on Advanced Computer Theory and Engineering IEEE Computer Society 2008, 902- 906.

4. F. Ricca and P. Tonells._Analysis and testing of Web applications. InternationalConference on Software Engineering Proceedings of the 23rd International Conference on Software Engineering. IEEE Computer Society 2001, 25-34.

5. Huai-kou Miao, Zhongsheng Qian and hongwei Zeng. A Practical Web Testing Model for Web Application Testing. Third International IEEE Conference on Signal Image Technologies and Internet-Based System.

6. K.K.Aggrawal, Yogesh Singh, A.Kaur Code Coverage Based Technique For Prioritizing Test Cases For Regression Testing ACM SIGSOFT Software Engineering September 2004 Volume 29 Number 5

7. Shengbo Chen, Huaikou Miao, Zhongsheng Qian Automatic Generating Test Cases for Testing Web Applications Third International IEEE Conference on Signal-Image Technologies and Internet-Based System

8. Srinivasan Desikan A test methodology for an effective regression testing

9. Alex Chaffee (2000-08-17). "What is a web application (or "webapp")?". http://www.jguru.com/faq/view.jsp?EID=129328. Retrieved 2008-07-27.

10. A. Andrews, J. Offutt, and R. Alexander, “Testing Web Applications by Modeling with FSMs”, *Software and Systems Modeling*. 2004.

11.Testing”,http://www.soft.com/products/web/technology/websitetesting.html.

12. G. A. D. Lucca and A. R. Fasolino, “Testing Web-based Applications: The State of the Art and Future Trends”,*Information and Software Technology*, vol. 48, 2006, pp.1172-1186

13. Software Testing by Kaner and Bach

14.Multiple (wiki). "Web application framework". *Docforge*. htp://docforge.com/wiki/Web_application_framework. Retrieved 2010-03-06.

15. A. Tarhini, H. Fouchal, and N. Mansour, Regression Testing Web Services-based applications, ACS/IEEE Int.Conf. on Computer Systems and Applications, 2006, pp.163-170.

16. N. Mansour, and M. Houri, Testing Web Applications,Information and Software Technology, 2006, 48(1), pp. 31-42.

17.K.K.Aggrawal, Yogesh Singh, A.Kaur, Code Coverage Based Technique for Prioritizing Test Cases For Regression Testing, September 2004 Volume 29 Number 5

18. http://en.wikipedia.org/wiki/Web_application

19. http://www.sitepoint.com/development-guide-success/

20 http://testinginterviewquestionsandanswers.com/what-is-regression-testing.html

21 http://en.wikipedia.org/wiki/Web_testing

22. Seven Principles of Software Testing Bertrand Meyer, ETH Zürich and Eiffel Software

23. Software Testing Fundamentals—Concepts, Roles, and Terminology John E. Bentley, Wachovia Bank, Charlotte NC

APPENDIX A

Appendix A

A.1 Code of the System

```
#include<iostream.h>
#include<conio.h>
#include<fstream.h>
#include<string.h>
#include<stdio.h>
#include<process.h>

struct student
{
	int id;
	char dt[10], mm[10], yr[10];
	char mob[10];
	char username[50], password[50];
	char type[50];
}s1,s2;
struct student2
```

```
{
        int id;
        char username[50];
        char type[50];

}s3;
int flag =0,flag1=0, flag2=0,flag3=0, flag4=0, flag5=0, flag6=0;
char
name[20],pwd[20],name2[20],pwd2[20],name3[20],pwd3[20],name4[20],pwd4[20];
int id1, id2,id3,id4;
void testlogin();
void testlogin_display();
void testforpass() ;
void testforget_display() ;
void testsignup() ;
void testsignup_display();
void check_uname_pwd();
void check_userpass() ;
void check_userpass2() ;
void check_userpass3() ;
void check_forgetpass();
void check_uname_pass1();
void check_uname_pass2();
void check_uname_pass3();
void testlogin()
{       clrscr();
        fstream fout, fin;
        cout<<"\n Test Case for Valid Username and valid Password \n";
        cout<<"\n Username :";gets(s1.username);
        strcpy(name,(s1.username));
        cout<<"\n Password :";gets(s1.password);
        strcpy(pwd,(s1.password));
```

```
//fout.write((char*)&s1, sizeof(s1));
        check_uname_pwd();
        if(flag != 1)
        {
        flag=1;
        cout<<"\n Test Case for valid Username and invalid Password \n";
        cout<<"\n Username :";gets(s1.username);
        strcpy(name2,(s1.username));
        cout<<"\n Password :";gets(s1.password);
        strcpy(pwd2,(s1.password));
        //fout.write((char*)&s1, sizeof(s1));
        check_userpass();
        if(flag2 != 1)
        {
        flag2=1;
        cout<<"\n Test Case for invalid username and valid password\n" ;
        cout<<"\n Username :";gets(s1.username);
        strcpy(name3,(s1.username));
        cout<<"\n Password :";gets(s1.password);
                strcpy(pwd3,(s1.password));
        //fout.write((char*)&s1, sizeof(s1));
        check_userpass2();

        if(flag3 != 1)
        {
        flag3=1;
        cout<<"\n Test Case for invalid username and invalid password\n";
        cout<<"\n Username :";gets(s1.username);
        strcpy(name4,(s1.username));
        cout<<"\n Password :";gets(s1.password);
        strcpy(pwd4,(s1.password));
        //fout.write((char*)&s1, sizeof(s1));
```

```
                check_userpass3();
                fout.close();
                }
                }
                }
}
        void testlogin_display()
        {
                testlogin();
                cout<<"\n Record Entered Successfully\n";
                //flag=1, flag2=1;
                fstream fin;
                fin.open("user3.dat",ios::in|ios::binary);
                cout<<"\n The following generated test cases for LOGIN functionality are
\n\n\n";
                fin.seekg(0L,ios::end);
                long int l=fin.tellg();
                int count = (l/sizeof(s1));
                fin.seekg(0L,ios::beg);
//              clrscr();
                cout<<"S.NO."<<"\t"<<"DESCRIPTION"<<"\t"<<"INPUT
DATA"<<"\t"<<"EXPECTED OUTPUT"<<"\t  "<<"ACTUAL OUTPUT\n";
                for(int i=0;i<count;i++)
                {
                        cout<<"\n\n";
                        cout<<i+1<<"\t";
                        switch(i+1)
                        {
                                case 1: cout<<"valid username \n \tvalid password";
                                fin.read((char*)&s1, sizeof(s1));
                                cout<<"\t "<<s1.username<<","<<s1.password; break;
                                case2:cout<<"validusername\n"<<"\t"<<"invalidp
```

```
password";
                                fin.read((char*)&s1, sizeof(s1));
                                cout<<" "<<s1.username<<","<<s1.password; break;
                                case 3: cout<<"invalid username \n \tvalid password";
                                fin.read((char*)&s1, sizeof(s1));
                                cout<<"\t "<<s1.username<<","<<s1.password; break;
                                case   4:   cout<<"invalid   username   \n"<<"\t"<<"invalid
password";
                                fin.read((char*)&s1, sizeof(s1));
                                cout<<" "<<s1.username<<","<<s1.password; break;
                        }
                        switch(i+1)
                        {
                                case   1:   cout<<"\tAcess   granted";        cout<<"\t   Acess
granted";break;
                                case 2: cout<<"\tAcess denied ";   cout<<"\t Acess denied";
break;
                                case 3: cout<<"\tAcess denied ";   cout<<"\t Acess denied";
break;
                                case 4: cout<<"\tAcess denied ";   cout<<"\t Acess denied";
break;
                        }

                }
                getch();

        fstream fout;
        fout.open("user3.dat",ios::out|ios::binary);
        fout.close();
        }

void check_uname_pwd()
```

```
{
   fstream fin,fout;
   flag2=0;
   flag3=0;
   int check=0;

        fin.open("user2.dat",ios::in|ios::binary);
        fin.seekg(0L,ios::end);
        long int l=fin.tellg();
        int count = (l/sizeof(s1));
        fin.seekg(0L,ios::beg);

        for(int i=0;i<count;i++)
        {
                  fin.read((char*)&s1, sizeof(s1));
                  if(strcmp(name,s1.username)==0)
                  {
                            if(strcmp(pwd,s1.password)==0)
                            {

                            cout<<"\nCorrect entries";
                            check =1; flag=0;
                            fstream ankit;
                            ankit.open("user3.dat",ios::app|ios::binary);
                            strcpy((s2.username),name);
                            strcpy((s2.password),pwd);
                            ankit.write((char*)&s2, sizeof(s2));
                            ankit.close();
                            }
                            else
                            {
                               cout<<"\n Valid username BUT INCORRECT password";
```

```
                        }
                }
                else
                {
                        //username entered is invalid
                }
        }
        fin.close();

        if(check==0)
        {

                cout<<"\nIncorrect entries";
                getch();
                //goto abc;
                        fstream fout;
        fout.open("user3.dat",ios::out|ios::binary);
        fout.close();

                testlogin();
        }

}




void check_userpass()
{
   flag3=0;
  fstream fin;
  int check1=0;
        fin.open("user2.dat",ios::in|ios::binary);
fin.seekg(0L,ios::end);
long int l=fin.tellg();
int count = (l/sizeof(s1));
fin.seekg(0L,ios::beg);

for(int i=0;i<count;i++)
{
        fin.read((char*)&s1, sizeof(s1));
        if(strcmp(name2,s1.username)==0)
        {
                if(strcmp(pwd2,s1.password)!=0)
                {
                cout<<"\nCorrect entries";
                check1 =1;
                fstream ankit;
                ankit.open("user3.dat",ios::app|ios::binary);
                strcpy((s2.username),name2);
                strcpy((s2.password),pwd2);
                ankit.write((char*)&s2, sizeof(s2));
                ankit.close();


                }
                else
                {
                //cout<<"incorrect entries";
                check1=0;
                //flag2=1;
                }

        }
        else
```

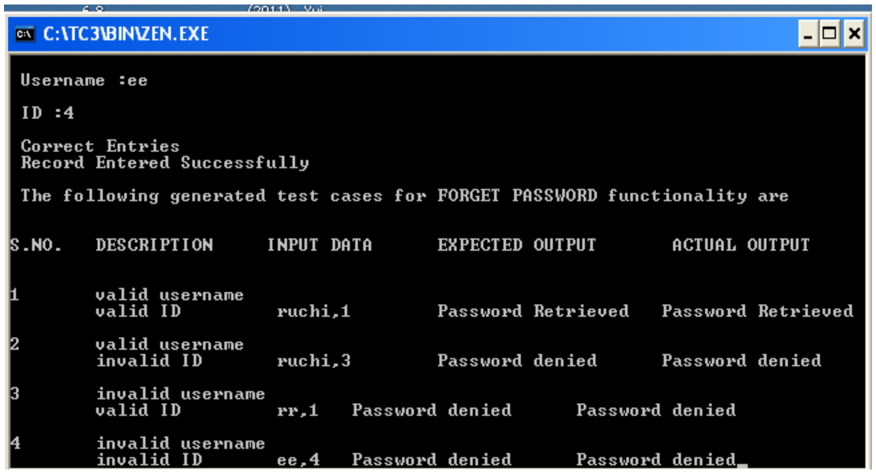
```
                {
                        //username entered is invalid

                }
        }
        fin.close();

        if(check1==0)
        {

                cout<<"\nIncorrect entries";
                //cout<<"\n flag2=1";
                getch();
                //goto abc;
                //flag2=1;
                        fstream fout;
        fout.open("user3.dat",ios::out|ios::binary);
        fout.close();

                testlogin();

        }

}



void check_userpass2()
{
   fstream fin;
   int check2=0;
        fin.open("user2.dat",ios::in|ios::binary);
        fin.seekg(0L,ios::end);
```

```
long int l=fin.tellg();
int count = (l/sizeof(s1));
fin.seekg(0L,ios::beg);

for(int i=0;i<count;i++)
{
        fin.read((char*)&s1, sizeof(s1));

        if(strcmp(pwd3,s1.password)==0)
        {
                if(strcmp(name3,s1.username)!=0)
                {
                cout<<"\nCorrect entries";
                check2 =1;
                fstream ankit;
                ankit.open("user3.dat",ios::app|ios::binary);
                strcpy((s2.username),name3);
                strcpy((s2.password),pwd3);
                ankit.write((char*)&s2, sizeof(s2));
                ankit.close();

                }
                else
                {
                check2=0;
                //flag2=1;
                }

        }
        else
        {
                //username entered is invalid
```

```
            }

        }
        fin.close();

        if(check2==0)
        {

            cout<<"\nIncorrect entries";
            //cout<<"\n flag3=1";
            getch();
            //goto abc;
            //flag3=1;
                fstream fout;
        fout.open("user3.dat",ios::out|ios::binary);
        fout.close();

            testlogin();
        }
}

void check_userpass3()
{
   fstream fin;
   int check3=1;
        fin.open("user2.dat",ios::in|ios::binary);
        fin.seekg(0L,ios::end);
        long int l=fin.tellg();
        int count = (l/sizeof(s1));
        fin.seekg(0L,ios::beg);
```

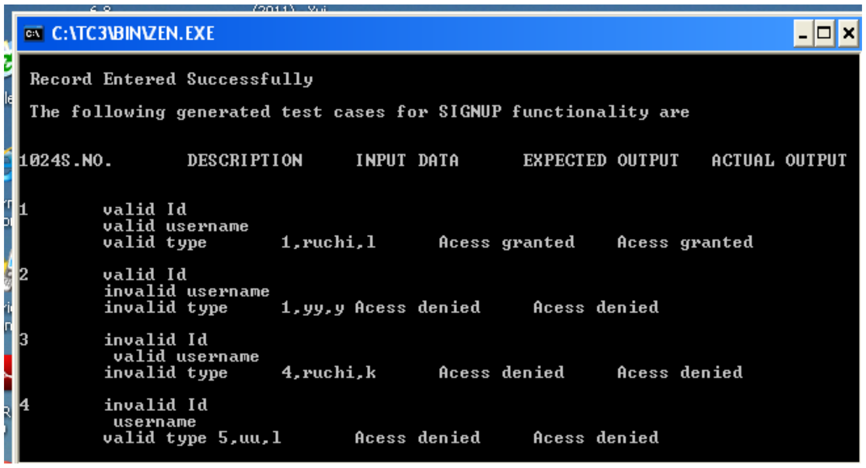
```
for(int i=0;i<count;i++)
{
        fin.read((char*)&s1, sizeof(s1));
        if(strcmp(name4,s1.username)==0)
        {       check3=0;
                if(strcmp(pwd4,s1.password)==0)
                {
                check3 =0;
                }
        }

}
fin.close();

if(check3==0)
{
        cout<<"\nIncorrect entries";
        getch();
        //goto abc;
                fstream fout;
fout.open("user3.dat",ios::out|ios::binary);
fout.close();

        testlogin();

}
else
{
        cout<<"\n Correct Entries";
        fstream ankit;
                ankit.open("user3.dat",ios::app|ios::binary);
                strcpy((s2.username),name4);
                strcpy((s2.password),pwd4);
                ankit.write((char*)&s2, sizeof(s2));
                ankit.close();

        }

}
```

A.2 Screenshots of system



```
Username :ee

ID :4

Correct Entries
Record Entered Successfully

The following generated test cases for FORGET PASSWORD functionality are

S.NO.   DESCRIPTION      INPUT DATA       EXPECTED OUTPUT       ACTUAL OUTPUT

1       valid username
        valid ID         ruchi,1          Password Retrieved    Password Retrieved

2       valid username
        invalid ID       ruchi,3          Password denied       Password denied

3       invalid username
        valid ID         rr,1    Password denied      Password denied

4       invalid username
        invalid ID       ee,4    Password denied      Password denied_
```

A.2.1   test Case for Login functionality



```
 Record Entered Successfully
 The following generated test cases for SIGNUP functionality are

1024S.NO.       DESCRIPTION     INPUT DATA      EXPECTED OUTPUT    ACTUAL OUTPUT

1        valid Id
         valid username
         valid type       1,ruchi,l      Acess granted     Acess granted

2        valid Id
         invalid username
         invalid type     1,yy,y Acess denied      Acess denied

3        invalid Id
          valid username
         invalid type     4,ruchi,k      Acess denied      Acess denied

4        invalid Id
          username
         valid type 5,uu,l       Acess denied      Acess denied
```

A.2.2 test Case for Login functionality