Review Paper on Various Software Testing Techniques & Strategies

Article *in* Global Journal of Computer Science and Technology · May 2019

DOI: 10.34257/GJCSTCVOL19IS2PG43

2 authors:

Nahid Anwar
Boise State University
2 PUBLICATIONS **32** CITATIONS
SEE PROFILE

Susmita Kar
Bangladesh University of Business and Technology
3 PUBLICATIONS **39** CITATIONS
SEE PROFILE

Global Journal of Computer Science and Technology: C
Software & Data Engineering
Volume 19 Issue 2 Version 1.0 Year 2019
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Review Paper on Various Software Testing Techniques & Strategies

By Nahid Anwar & Susmita Kar

Bangladesh University of Business and Technology

Abstract - Software testing is the process of running an application with the intent of finding software bugs (errors or other defects). Software applications demand has pushed the quality assurance of developed software towards new heights. It has been considered as the most critical stage of the software development life cycle. Testing can analyze the software item to identify the disparity between actual and prescribed conditions and to assess the characteristics of the software. Software testing leads to minimizing errors and cut down software costs. For this purpose, we discuss various software testing techniques and strategies. This paper aims to study diverse as well as improved software testing techniques for better quality assurance purposes.

Keywords: testing techniques, testing tools, verification, validation, level of testing, debugging, software testing objectives, software testing principles, software testing strategies, debugging, testing methodologies, software testing life cycle.

GJCST-C Classification: D.2.5

Strictly as per the compliance and regulations of:

Review Paper on Various Software Testing Techniques & Strategies

Nahid Anwar[α] & Susmita Kar[σ]

Abstract- Software testing is the process of running an application with the intent of finding software bugs (errors or other defects). Software applications demand has pushed the quality assurance of developed software towards new heights. It has been considered as the most critical stage of the software development life cycle. Testing can analyze the software item to identify the disparity between actual and prescribed conditions and to assess the characteristics of the software. Software testing leads to minimizing errors and cut down software costs. For this purpose, we discuss various software testing techniques and strategies. This paper aims to study diverse as well as improved software testing techniques for better quality assurance purposes.

Keywords: testing techniques, testing tools, verification, validation, level of testing, debugging, software testing objectives, software testing principles, software testing strategies, debugging, testing methodologies, software testing life cycle.

I. Introduction

Software development involves developing software against a set of requirements. Software testing is needed to verify and validate that the software that has been built to meet these specifications. Software testing helps in the prevention of errors in a system. It refers to the process of evaluating the software to find out the error in it. It is also used to analyze the software for other aspects of the software like usability, compatibility, reliability, integrity, efficiency, security, capability, portability, maintainability, etc. Software testing aims at achieving specific goals and principles which are to be followed. In simple words, testing is the process of locating errors in the program. Software Testing is executing the software to (i) perform verification, (ii) to detect the mistakes and (iii) to achieve validation.

i. *Verification:* It is the process of checking the software concerning the specification. [Verification: Are we making the product, right?]
ii. *Error Detection:* It is the process of deliberately performing the wrong inputs to check the system's performance.
iii. *Validation:* It is the process of checking software concerning the customer's expectation. [Validation: Are we making the right product?]

Author α σ: Department of Computer Science and Engineering Bangladesh University of Business and Technology.
e-mails: nahidanwar007@gmail.com, susmitak275@gmail.com

Testing is characterized as a process of assessment that either the definitive system meets it's specified fulfillments initially or not. It is mainly a system beset with the validation and verification process that whether the developed system meets the fulfillments defined by the customers. Therefore, this activity draws the difference between the actual and expected result. So, this is an analysis that provides the associates with the proper knowledge about the quality of the product. Software Testing can also be defined as a risk-based activity. The testing cost and errors can be found in a relationship in figure 1. It is apparently demonstrated that cost rises dramatically in a testing (functional and nonfunctional). The compelling testing goal is to do the optimal amount of tests so that additional testing endeavor can be minimized. From Figure 1, we can say that Software testing is an essential factor in software quality assertion.

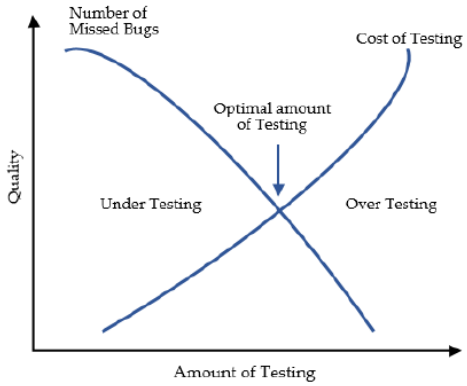

Figure 1: Every Software Project has optimal test effort (Courtesy)

a) Basic Terminology

Mistake, Error: Human makes a mistake. A suitable synonym is an error. It differences between the actual output and the expected output.

Bug: It means that when a developer makes the error while coding. It is the state which is responsible for the failure of the specific function.

Fault: It is the representation of the error, where representation is the mean of expression that may be diagrams, flow charts, etc.

Failure: Failure occurs when a fault is executed.

b) Software Testing Objectives

The fundamental objective of software testing is to provide a quality product regarding the reliability estimation and complete verification and validation of the product. The secondary aim of testing includes executing a program with the intent of finding errors and producing a test case which is capable of detecting the undiscovered error as yet.

II. Literature Review

In this section, we will outline the previous works of Software Testing. According to "The Theory of Software Testing", testing is the means of showing the presence of errors in the program which can either be performed manually or automatically. It also includes the basic terminology of testing such as automated testing, failure, testing team, and wrong test case selection. This paper focuses on the process that should be followed to test the performance of new software and the entire system. The conclusion of the article is the complete view of software testing, preliminary testing, and user acceptance testing.

III. Software Testing Methodologies

The importance of software testing to software quality cannot be overemphasized. After the development of the code, it is mandatory to test the software to identify all the errors, and they must be debugged before the release of the software. Although it is impossible to identify and debug all the errors in the significant software at every phase, it is tried to remove all the mistakes as possible. Testing helps in finding the bugs; it cannot conclude that the software is bug-free. We broadly categorized testing techniques into two parts:

1. Static Testing
2. Dynamic Testing

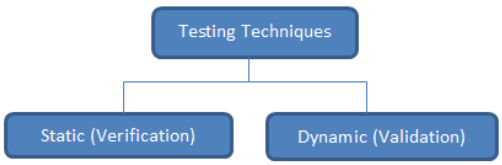

Figure 2: Testing Techniques

1. *Manual Testing (Static Testing):* It refers to the method of testing where the code is not executed. It does not require highly skilled professionals since the actual execution of the system is not done in this process. It starts with the initial phase of the Software Development Life Cycle (SDLC); hence, it is also known as verification testing. The main objective of static testing is to enhance the quality of software products by helping software professionals to Identify and resolve their errors early in the software development process. Static testing is performed on the documents like Software Requirement Specification (SRS), design documents, source code, test suites, and web page content. It is performed before code deployment. As a result, it provides the evaluation of code as well as documentation. Static testing techniques include:
   i. *Inspection:* It is primarily done to locate defects. Moderators conduct the code walkthrough. In this type of formal review, a checklist is prepared to check the working document.
   ii. *Walkthrough:* It is not a formal process. The authors lead this process. The Author advises the participants through the document according to his or her thought process so that they can accomplish a common perception. It is especially useful for higher-level documents like requirement specification, etc.
   iii. *Technical Reviews:* A professional round of review is operated to check if the code is made in consonance with the technical specifications and norms which may include the test plans, test strategy, and test scripts.
   iv. *Informal Reviews:* It's the static testing technique in which the document is reviewed unofficially, and helpful comments are implemented.
2. *Automated Testing (Dynamic Testing):* Dynamic Testing is a kind of software testing technique in which the dynamic behavior of the code is analyzed. In dynamic testing, also known as validation where the actual system is considered. It requires the highly skilled professional with the proper domain knowledge. Dynamic testing involves testing the software for the input values, and output values are analyzed. Progressive testing is divided into two categories:
   A. White Box Testing
   B. Black Box Testing
   C. Grey Box Testing

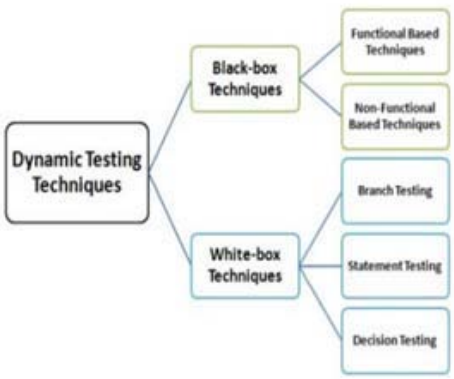

Figure 3: Dynamic Testing Techniques

a) *White Box testing*

Internal specifications and structures of the system are created conspicuous. So, it's acutely cost-effective in detecting and resolving problems. Bugs will be found before they cause bickering. Thus, we will summarize this approach as testing software with the data of its internal structure and coding. White box testing is also familiar as precise box analysis or white box analysis or glass box testing or transparent box testing, and structural testing. It's an approach for finding errors within which the tester has complete data. This technique isn't used much for debugging in large systems and networks. Different types of white box testing include basis path, loop testing, control structure testing, etc. White-box testing tests internal constructions or workings of a program because programming skills and the domestic context of the system are used to design test cases. The tester appoints inputs to apply paths through the code and finalize the appropriate outputs. This is akin to testing nodes in a circuit. White-box Testing can be used at the unit, integration and system levels of the software testing process. It usually is done at the unit level. Although this approach of testing can expose many errors, it may not identify the missing requirements and unimplemented parts of the specification.

White-box Testing includes the following approaches:

- Application Programming Interface testing tests the application using public and private APIs by creating tests to satisfy some criteria for code coverage.
- Fault Injection Methods – Introducing faults to gauge the efficacy of testing strategies intentionally.
- Code coverage tools can assess the integrity of a test suite that was created with any method, including black-box testing. This grants the software team to check the parts of a system that are rarely tested and assures that the most important function points have been verified.
- Function coverage is the approach which informs on functions that have been executed.
- Statement coverage is the approach which informs the number of lines executed to complete the test 100 per cent. It assures that all code paths or branches are executed at least once. This helps to ensure correct functionality.

Advantages:

1. It exposes an error that is hidden in code by eliminating extra lines of code.
2. Maximum coverage is obtained during test outline writing.
3. The developer discreetly gives reasons for implementation.

Disadvantages:

1. An experienced tester is required to carry out this procedure because knowledge of internal structure is needed.
2. Many paths will remain untested since it is challenging to look into every pros and con.

b) *Black Box testing*

A black box testing is a testing in which internal specifics and workings aren't known or accessible to its customer. It supports specifications and output needs. The fundamental purpose is to identify the requirements of the system. Black box testing has very little or no data on the inner logical structure of the system. Thus, it solely checks the basic features of the system. It assures that every input is appropriately accepted and outputs are correctly produced. Black-box testing checks the functionality without any knowledge of the internal implementation. The testers only have an understanding of what the software is supposed to do, not how it does it. This is solely done based on customers' aspect. The tester only knows the set of inputs and specific outputs.

Black-box testing methods include:

i. *Equivalence Partitioning:* This technique divides the input domain of a program into equivalent classes from which test cases can be derived. Thus, it can minimize the number of test cases.
ii. *Boundary Value Analysis:* It targets the testing at borders, or where the extreme boundary values are chosen. It comprises of minimum, maximum, error values, and typical values.
iii. *Fuzzing:* This approach takes random input to the application. It is used for characterizing implementation bugs, using malformed or semi-malformed data injection in an automated or semi-automated session.
iv. *Orthogonal Array Testing:* In this technique, the input domain is minimal but too large to accommodate exhaustive testing.
v. *Cause-Effect Graph:* This testing technique begins by generating a graph and establishing the relation between effect and its causes.
vi. *All Pair Testing:* The main objective is to have a set of test cases that covers all the pairs. Here, test cases are designed to execute all possible discrete combinations of each couple of input parameters.
vii. *State Transition Testing:* This testing approach is useful for the navigation of a graphical user interface.

Advantages:

1. Testers do not need to know specific programming languages. Testing is done from a user's point of view.
2. It helps to find out any ambiguities or inconsistencies in the requirement specifications.

3. Programmer and validators both are independent of each other.

Disadvantages:

1. Test cases are difficult to design without fair stipulations.
2. Probability of having the repetition of tests that are already done by the programmer.
3. Here, some parts of the back end are not tested at all.

c) *Grey-Box Testing*

Gray box testing is the technique of testing the software with limited knowledge of the internal structure and design of the application. It is defined as a testing software package which has some data of its internal logic and underlying code. It uses internal information structures and algorithms. This approach holds necessary conducting integration testing between two or more modules of code written by totally different developers. This approach includes reverse engineering to work out on the boundary values. Grey box testing is unbiased and non-intrusive.

Grey-box Testing has the knowledge of internal data structures and algorithms for purposes of designing tests while executing those tests at the user level. The tester does not have full access to the software's source code. The following are some subtypes of grey-box testing:

i. *State-Model-Testing:* It examines each method of an object, transition & transition paths at each state of an object.
ii. *Class-Diagram Testing:* It examines all the derived classes of the base class.
iii. *Sequence-Diagram Testing:* It examines all the methods occurring in the sequence diagram.
iv. *Thread-Based Testing:* In this approach, all the classes of single Use-Cases are integrated and then testing is performed. This approach is going on until all the courses of all Use-Cases have been considered.
v. *Use-Based Testing:* In this testing, the testing on the classes that either needs the services from other courses or does not need any services are performed.

Advantages:

1. It implements the combined benefits of black-box and white-box testing techniques.
2. In grey box testing, the tester can design high test scenarios.
3. Testing is unbiased.

Disadvantages:

1. Test coverage is limited because the access to source code is not available.
2. Many program paths remain untested.
3. The test cases can be redundant.

3. *Nonfunctional Testing Techniques:* It is a type of testing which is performed to test various attributes of a system like stress, load, etc. Nonfunctional testing is performed at all test levels. It is concerned with the nonfunctional requirements and is explicitly modeled to assess the readiness of a system according to the various criteria which are not covered by functional testing.

Figure 4: Non-functional Testing Techniques

a) *Performance Testing:* It assesses the entire performance of the system. It is used to evaluate the system's performance under a specific workload.
b) *Load testing:* A load test is performed to ensure the load taking the capacity of a particular order. Load testing is performed to determine the behavior of the system under normal as well as peak load conditions.
c) *Endurance Testing.* It is the type of testing performed to determine the system's behavior after a particular time. For example, a method is working perfectly fine in the initial first hour, but the performance decreased after three hours of execution.
d) *Stress Testing:* It is performed to determine the system performance beyond standard operational capacity, often to a breaking point. It is concerned with the system's load taking ability.
e) *Security Testing:* Security testing is done to assess that the system is safe or not. It is a process that is concerned with the fact that an information system protects data and maintains functionality as intended.
f) *Recovery Testing:* Recovery testing is performed to check the recovery of the system after the crash or hardware failure. In this type, the software is forced to fail under a given circumstance and then finally, the recovery is tested.
g) *Compatibility Testing:* Compatibility testing is concerned with checking the system's compatibility with the rest of the environment. It checks the

harmony of the developed system with the various other components such as hardware, additional software, DBMS and operating system, etc.

IV. Software Testing Strategies

Software Testing strategies provide a method of integrating software test case design methods into a well-planned Series of steps that can result in the successful construction of software. It provides the road map for testing. The software testing Strategy should be pliable enough to develop a customized testing approach. The software testing strategy is actually produced by project managers, software engineers, and testing specialists. There are four different types of software testing strategies:

1) Unit testing
2) Integration testing
3) Acceptance/Validation testing
4) System testing

1. *Unit testing*

Unit is the smallest testable part, i.e. the most modest collection of lines of code which can be tested. Unit testing is done by the developer as the proper knowledge about the core programming designing is required. Generally, unit testing is considered as a white-box testing class because it is partisan to evaluate the code as implemented rather than assessing conformance to some set of requirements.

Benefits of Unit Testing:

1) Cost-effective testing technique.
2) Simple testing technique because the smallest testable unit of the code is tested here.
3) Individual parts are tested when necessary, without waiting for another part of the system.
4) Unit testing can be performed in parallel by fixing problems simultaneously by many engineers.
5) Detection and removal of defects are much cost-effective compared to other levels of testing.
6) Be able to take advantage of several formal testing approaches available for unit testing.
7) Clarify debugging by limiting to a small unit the possible code areas in which to search for bugs.
8) Be able to test internal logic that is not easily reached by external inputs in the broader integrated systems.
9) Attain a high level of structural coverage of the code.
10) When debugging severe problems, it avoids lengthy compile-build-debug cycles.

Unit testing techniques: Unit testing uses several effective testing techniques. The testing techniques categorize into three types:

i. Functional Testing
ii. Structural Testing
iii. Heuristic or Intuitive Testing

2. *Integration testing*

Integration testing is an efficient technique for constructing the program structure as well as to perform tests to uncover errors related to interfacing. The objective of integration testing is to integrate the unit tested component and tested them as a group. Different Integration testing Strategies are:

a) Top-down Integration testing
b) Bottom-up Integration testing

a) *Top-down Integration:* It is an incremental procedure to build a program format. Formats are integrated by moving downward through the fabric, beginning with the main control module. Modules subordinate to the central control module are incorporated into the structure in either a depth-first or breadth-first manner. The integration activity is performed in a series of five steps:
   i. The main control format is used as a test driver, and stubs are substituted for all components directly subordinate to the central control module.
   ii. Depending on the integration techniques, selected subordinate stubs are replaced one at a time with actual parts.
   iii. Tests are conducted as each element is integrated.
   iv. On completion of each set of experiments, another stub is replaced with the real component.
   v. Regression testing may be managed to ensure that new errors have not been introduced. It is not as relatively straightforward as it looks. In this logistic problem can arise. The problem occurs when testing a low-level module that requires checking the upper level. Stub replaces low-level modules at the beginning of top-down testing. So no data can flow in an upward direction.

b) *Bottom-up Integration:* It begins development and testing with atomic modules. Since modules are integrated from the bottom-up, processing required for elements subordinate to a required level is always available. The following steps are implemented with bottom-up integration strategy:
   i. Low-level components are combined into clusters so that they can perform a specific software sub-function.
   ii. A driver is reported to coordinate test case input and output.
   iii. The group is tested.
   iv. Drivers are detached, and clusters are combined moving upward in the program structure.

3. *Acceptance testing*

In this approach, testing is carried out to authenticate whether the product is developed as per the standards and detailed criteria and meets all the requirements specified by the user. The user carries this

type of testing where the product is developed externally by the third party. Acceptance testing falls under the black-box testing approach, where the user is not very much involved in the internal working of the scheme. Acceptance testing approach is also known as validation testing, QA testing, final testing, factory acceptance testing and application testing, etc. In software engineering, acceptance testing may be executed at two different levels; one at the system provider level and another at the end-user level.

Classification of Acceptance Testing:

i. *User Acceptance Testing:* User acceptance testing is an essential step before the system is finally deployed to the end-user. User acceptance testing is generally done by the actual software uses to ensure that it can handle the specified task in the real world scenarios.

ii. *Alpha Testing & Beta Testing:* QA teams or developers generally have done this type of testing. Unit testing, Integration testing and System testing combine name as alpha testing. Alpha testing is conducted in the presence developers and in the absence of users. In this testing, the following criteria are examined such as spelling mistakes, broken lines, cloudy direction etc.

After completing the alpha testing successfully, beta testing is performed. Beta testing is conducted by real users who actually handle the software in real-world scenarios. The customers provide their assessment to the developer for the outcome of the experiment. It is also known as field testing. Feedback from the users is used to improve the performance of the system/product before it is released to other users/customers.

iii. *Operational Acceptance Testing:* Operational acceptance testing (functional preparedness testing) is an approach of assuring all the specified processes and procedures of the system are in place to allow user/tester to use it.

iv. *Contact and Regulation Acceptance Testing:* The system is tested against the required criteria as mentioned in the contract document and also proved to check if it meets all the government and local authority rules and regulations, even all the essential standards.

4. *System testing*

System testing is a level of software or hardware testing where testing is conducted on a complete, integrated system to assess the system's compliance with its specified requirements. System testing falls within the category of black-box testing. The purpose of system testing is to evaluate the system compliance against the specified requirements.

Some of the Different types of system testing are as follows:

i. Recovery testing
ii. Security testing
iii. Graphical user interface testing
iv. Compatibility testing

i. *Recovery Testing:* In this process of testing an application is able to recover from crashes, hardware failures and other similar problems. If recovery is automatic, re-initialization, check-pointing mechanisms, data recovery, and restart are evaluated for correctness. Recovery testing force the software to fail in a variety of ways to assess that reconstruction is performed correctly.

ii. *Security testing:* Security testing tries to evaluate those protection mechanisms built into a system will protect it from erroneous penetration. During security testing, the tester individually desires to penetrate the system. The validator may try to acquire passwords through external clerical means; may attack the order with custom software designed to breakdown any defenses that have been constructed; may overwhelm the system, thereby denying service to others; The goal of security testing is to identify the threats in the system and measure its potential vulnerabilities.

iii. *Graphical user interface testing:* In this testing approach, the product's graphical user interface is involved to ensure it meets its written specifications. GUI testing attempts to check the screens with the controls like menus, buttons, icons, and all types of bars - toolbar, menu bar, dialogue boxes, and windows, etc.

iv. *Compatibility testing:* Compatibility testing is concerned with checking the system's compatibility with the rest of the environment. It checks the harmony of the developed system with the various other components such as hardware, additional software, DBMS and operating system, etc.

V. Discussion

In this section, the difference between testing and debugging is shown. Software testing is a process of executing a program in order to identify whether it meets the specified requirement or not. The test case design is conducted, a strategy can be defined, and results can be evaluated against prescribed expectations. Debugging is the process of fixing the problems that are found during the software testing process. It is a consequence of successful testing. That is when testing uncovers a defect, debugging is the process that leads to the removal of the error. The debugging process can be done by the programmer or the developer. Based on the defect that is reported, the developers try to find the root causes of the error to

make the system defect free. Debugging is done in the three phases of software development. The first one is during the coding process when the programmer translates the design into executable code. In the coding process, the errors that are made by the programmer when writing in code needs to be detected quickly and fixed before going to the next phase of development. In these cases, the developer performs unit testing to identify any defects at the module or component level. Then debugging is performed during the last stage of testing. These involve debugging multiple components or a complete system, when unexpected behavior such as wrong return codes or abnormal program termination may be found.

VI. Conclusion

To conclude our survey, we would like to find that Software testing is an essential activity of the Software Development Life Cycle (SDLC). We can never say that a product is "Perfect". Testing is a never-ending process. Testing only shows the presence of errors, not the absence. It is time-consuming and an intensive process, therefore, upgraded techniques and innovative methodologies are necessary to maintain the quality of the software. This leads to performing Automated Testing implementation before and during the testing process. This paper aims to describe in detail various software testing techniques, the need for software testing, software testing goals, and objectives. Software testing is often less formal and meticulous than it should. In order to perform software testing effectively and efficiently, those who are involved in testing must be familiar with the basic concept, goals and principle of software testing.

We further discuss different software testing techniques such as white-box Testing, black-box Testing, grey-box Testing, Security testing etc. Hence, the future work in relevance with the testing process will be much more technology-dependent harnessing the simulation and automated testing model-based approach, not only expediting the testing life cycle but also providing optimum bug prevention and efficient quality assurance.

References Références Referencias

1. Kaner, Cem (November 17, 2006). "Exploratory Testing" (PDF). Florida Institute of Technology, Quality Assurance Institute Worldwide Annual Software Testing Conference, Orlando, FL. Retrieved November 22, 2014.
2. Glenford J. Myers, "The Art of Software Testing, Second Edition" Published by John Wiley & Sons, Inc., Hoboken, New Jersey. 1.1. P. Mathur, "Foundation of Software Testing", Pearson/Addison Wesley, 2008.
3. P. Mitra, S. Chatterjee, and N. Ali, "Graphical analysis of MC/DC using automated software testing, in Electronics Computer Technology (ICECT), 2011 3rd International Conference on, 2011, vol. 3, pp. 145 –149.
4. F. Saglietti, N. Oster, and F. Pinte, "White and grey-box verification and validation approaches for safety- and security-critical software systems," Information Security Technical Report, vol. 13, no. 1, pp. 10–16, 2008.
5. "Software Testing Techniques and Strategies" Abhijit A. Sawant1, Pranit H. Bari2 and P. M. Chawan3 Department of Computer Technology, VJTI, University of Mumbai, INDIA.
6. Mohd. Ehmer Khan, "Different Forms of Software Testing Techniques for Finding Errors", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 1, May 2010.
7. Ajay Jangra, Gurbaj Singh, Jasbir Singh, and Rajesh Verma, "EXPLORING TESTING STRATEGIES", International Journal of Information Technology and Knowledge Management, Volume 4, NO.1, January-June 2011.
8. Software testing, by Ron Patton
9. Software engineering, by Roger Pressman
10. IEEE Standard 829-1998, "IEEE Standard for Software Test Documentation", pp.1-52, IEEE Computer Society, 1998.
11. Jovanovic and Irena, "Software Testing Methods and Techniques", May 26, 2008.