International Journal of Pure and Applied Mathematics
Volume 118 No. 24 2018
ISSN: 1314-3395 (on-line version)
url: http://www.acadpubl.eu/hub/
Special Issue

http://www.acadpubl.eu/hub/

AN ANALYSIS OF TESTING PROCESS, TYPES AND TECHNIQUES IN SOFTWARE ENVIRONMENT

#1V Swapna, Assistant Professor, Dept of CSE,VITS
#2M.Kishore Kumar, Associate Professor, Dept of CSE,VITS
JNT UNIVERSITY, Hyderabad.
#1swapnagourishetty52@gmail.com
, #2Kishore.mamidala@gmail.com

April 27, 2018

Abstract

Software engineering grasps a few controls dedicated to avert and cure breakdowns and to warrant sufficient conduct. Testing, the subject of this paper, is an across the board approval approach in industry, yet it is still to a great extent impromptu, costly, and capriciously powerful. In fact, software testing is an expansive term including an assortment of exercises along the improvement cycle and past, went for various objectives. Henceforth, software testing research faces a gathering of difficulties. A steady guide of the most pertinent difficulties to be tended to is here proposed. In it, the beginning stage is constituted by some imperative past accomplishments, while the goal comprises of four recognized objectives to which inquire about at last tends, however which stay as inaccessible as dreams. The courses from the accomplishments to the fantasies are cleared by the exceptional research challenges, which are talked about in the paper alongside intriguing continuous work. This paper discusses guaranteeing the nature of a wide range of software

applications by playing out specific kinds of testing strategies and streamlined software testing forms. According to the examination and research done testing composes can be ordered under three noteworthy testing systems which are Functional, Performance and Security Testing and real software testing process called as Analysis, Preparation and Execution and conclusion.

***Key Words*:**Functional, Performance and Security Testing (FPS), Analysis, Planning and Preparation, Execution and Closure (APEC), Software Testing Techniques, Software Testing Life Cycle (STLC), Software Development Life Cycle(SDLC).

1 INTRODUCTION

A challenge for IT industry is to create software framework that addresses business issues. The truth of the matter is we are to convey software that is free of bugs. The bugs in software can cause significant misfortune in IT association in the event that they are not evacuated before conveyance. Software testing is vital parameter creating software that is free from bugs and imperfections. Software testing is performed to help quality confirmation [5]. A decent quality software can be made by utilizing a productive test strategy. Measurements say that half of the aggregate cost of software improvement is given to software testing even it is more in the event of basic software[2]. Contingent upon time, scale and performing techniques we can arrange testing as unit testing, combination testing, framework testing, alpha testing, beta testing, acknowledgment testing, relapse testing, transformation testing, execution testing, push testing and so forth. There are trying like factual testing which is utilized for estimating unwavering quality of software as opposed to discovering blunders. Test information can be created either in view of particular [5, 6, 4] or code. In the writing of mechanized test information age, sought based information age review is available[6]. Despite the fact that code based overview for test information age has been talked about by numerous creators [9, 13] yet there is a field to ponder in setting of program analyzers, test information age models and so forth. In this paper we fundamentally focus on review of code based [9] test information age.

Test information can be planned either physically or consequently. Software building research puts substantial accentuation on computerizing the product improvement process that create huge more unpredictable amounts of code with less exertion. For testing these product, we have to discover progress inventive help strategies to robotize the testing process[13]. Regardless of nonstop exertion till today robotized testing has constrained effect in industry, where the test age movement remains to a great extent manual. What we require is 100% mechanized testing to diminish general cost of software advancement with high caliber. Various test information age procedures, for example, arbitrary test information generator, way situated test information generator, objective arranged test information generator and insightful test information generator have been robotized. These days testing on systems administration condition i.e. to enhance the versatility of software testing is accentuated [13].

Need of Software Testing:

Software development involves creating software against an arrangement of prerequisites. Software testing is expected to confirm and approve that the product that has been assembled has been worked to meet these details. If not we may most likely free our customer. So keeping in mind the end goal to ensure, that we give our customer an appropriate software arrangement, we go for testing [1]. Testing guarantees that what you get at last is the thing that you needed to construct. We look at if there is any issue, any mistake in the framework, which can make software unusable by the customer. This aides in the aversion of mistakes in a framework.

Goals For Software Testing:

Goals are the output of the software process. Software testing has following goals. [2]

1. Verification And Validation Testing can likewise be utilized for checking that the item or the product fills in as wanted and approve whether the product satisfies condition set down

2. Priority Coverage Testing ought to be performed in productive and viable way inside the financial plan and calendar limits.

3. Balanced Testing process must adjust the prerequisites, specialized impediment and client desire.

4. Traceable Documents ought to be set up of both achievement and disappointments of testing process. So no compelling reason to test same thing once more.

5. Deterministic We should comprehend what we are doing, what we are focusing on, what will be the conceivable result.

2 RELATED WORK

In the mid 1990s, there was an expanding enthusiasm for assessing and anticipating the unwavering quality of programming frameworks. Before Jalote and partners' investigation in 1994 [11], numerous current unwavering quality models utilized utilitarian testing procedures and anticipated the dependability in view of the disappointment information saw amid testing. The utilization of these models requires a considerable lot of information accumulation, calculation, and skill and calculation for deciphering the outcomes. The creators propose another approach in view of the scope history of the program, by which a product framework is displayed as a chart, and the dependability of a hub is thought to be a component of the circumstances it gets executed amid testing-the bigger the circumstances a hub gets executed, the higher its unwavering quality. The dependability of the product framework is then registered through recreation by utilizing the reliabilities of the individual hubs. With such a model, scope investigation devices can without much of a stretch be reached out to process the unwavering quality likewise, in this way completely mechanizing dependability estimation. The year 1997 saw great outcomes in both utilitarian and basic testing systems. In this year a structure for probabilistic useful testing was proposed. Bernot and associates present the detailing of the testing movement, which ensures a specific level of certainty into the rightness of the framework under test, and gives gauge of the unwavering quality [1]. They likewise clarify how one can create suitable dispersions for information spaces including most regular areas, for example, interims of whole numbers, associations, Cartesian items, and inductively characterized sets. Another fascinating examination in 1997 utilized formal engineering depiction for thorough, automatable strategy for joining trial of complex frameworks [4]. In this paper the creators propose

to utilize the formal detail dialect CHAM to show the conduct of enthusiasm of the frameworks. Diagram of all the conceivable practices of the framework as far as the collaborations between its parts is determined and additionally lessened. An appropriate arrangement of lessened diagrams features particular design properties of the framework, and can be utilized for the age of incorporation tests as per a scope methodology, comparable to the control and information stream charts in basic testing. This exploration is among the pattern of utilizing formal strategies in testing systems since later 1980s.

From late 1990s, Commercial Off The Shelf (COTS) programming and Unified Modeling Language (UML) UML have been utilized by expanding number of programming engineers. This pattern being developed accordingly calls the relating testing strategies for the UML parts.

In their 2000 paper [7], Hartmann and associates at Siemens tended to the issue of testing parts by incorporating test age and test execution innovation with business UML displaying instruments, for example, Rational Rose. The creators display their way to deal with demonstrating parts and associations, depict how test cases are gotten from these segment models and after that executed to check their conformant conduct. The TnT condition of Siemens is utilized to assess the approach by illustrations. Experiments are gotten from explained Statecharts. Another latest research is additionally a way to deal with test segment based programming.

In 2001, Beydeda and associates proposed a graphical portrayal of part based programming stream diagram [3]. Testing is influenced convoluted with highlights, for example, the nonappearance of segment to source code, that are particular to segment based programming. The paper proposes a method joining both black-box and white-box techniques. A graphical portrayal of part programming, called segment based programming stream diagram (CB-SFG), which imagines data accumulated from both determination and usage, is depicted. It would then be able to be utilized for experiment distinguishing proof in light of surely understood basic procedures.

There are ponders led to decide the best testing rehearses, for example, the one by Ram Chillarege, IBM (Chillarege, 1999) and Antonia (Bertolino, 2007). These examination thinks about present

extremely critical measure of information on great testing rehearses, be that as it may they are for the most part in view of hypothetical angles. This examination will attempt to answer comparable research inquiries with help of experimental information gathered through a modern contextual analysis. Additionally an overview was led in 2004 to examine the product testing hones in Australia by Reed. K. et al. (2004), which gave great bits of knowledge of programming testing hones valuable to outline this examination contemplate. Another as of late distributed research consider by Sundmark et al (2010) presents consequences of a mechanical study on contemporary parts of programming testing utilizing subjective and quantitative strategies. Their investigation gives significant data about inconsistencies saw between the flow hones and the impression of respondents which could demonstrate valuable in forming future research on programming testing, however the clarifications for these watched disparities were given in view of scientists suspicions, or at times the clarifications were not clear. For this situation think about the watched designs in view of respondents will be introduced and a clarification for the watched peculiarities or errors will be clarified by utilizing subjective information.

There are few research work led as of late to consider the appropriation of ISO 26262, for example, by Schwarz Juergen et al. (2009), Krammer, M et al. (2010), Lemmer, K et al. (2010), Matheis Johannes et al. (2010); however these examinations are not particular to programming testing rehearses. In this proposal we think about comparable perspectives yet concentrating on the product testing field.

3 OPTIMIZED SOFTWARE TESTING PROCESS

STLC stages manages recognizing and correcting any mistake by utilizing different programming testing strategies. This paper introduces the required periods of testing lifecycle without which no product life cycle would be finished effectively. Testing essentially outfits a feedback or an examination that decides the state conduct of the framework against its particulars, systems, standards, qualities and pertinent benchmarks. Programming testing proce-

dure can be redone as indicated by the client or the undertaking needs. The improvement procedure which one can utilize while testing programming is examination, arranging and readiness, execution and closure[10]. The product procedure gives the stream of the framework and upgrades the confirmation of the item to be created. There are different techniques for testing of programming that can be alluded from various research diaries, books and distributed papers however in light of study, look into and considering all the basic testing composes, this paper discuss the key discoveries that Functionality, Performance and Security testing are three principle programming strategies that a product analyzer should be tried to furnish programming as per determinations and with great quality.

3.1 Test Analysis Phase

The principal stage which is an Analysis stage is the fundamental period of the product testing process. This stage incorporates the examination of practical and non useful prerequisites e.g. business necessities, practical particular record and specialized detail archive etc[8]. The necessities gathering and is to be improved the situation clarification with clients to distinguish genuine and expected aftereffects of testing like Identification of prerequisites and holes, which are fundamentally non utilitarian prerequisites, for example, ease of use, versatility, testability, practicality, execution and security. All necessities that can't be tried because of framework and test condition requirements ought to be imparted to the business group. Amid this stage, the testing group surveys and investigations the necessities and recognizes the tests, which are to be performed and sets needs to test - colleagues. The test condition prerequisite incorporates the equipment and programming necessities under which the required programming is to be tried and in parallel programming designers begin by arranging and advancement exercises.

3.2 Test Planning and Preparation Phase

The test readiness stage incorporates test design planning, experiment, test information and test condition arrangement. The test design is the primary record to be readied, which traces the ex-

tension, targets, highlights to be tried, highlights not to be tried, sorts of testing to be performed, parts and obligations of testing group, passage and leave criteria and assumptions[1]. At the same time the testing groups begin getting ready experiments and test information. An experiment is a record, which diagrams steps required to test any usefulness with expected and genuine outcome. In the event that genuine outcome doesn't matches with expected outcome, at that point a bug is opened. For every necessity, positive and negative experiments are readied, which is guaranteed by Requirement Traceability Matrix (RTM). RTM is a report which maps necessities with test cases to guarantee 100% testing is finished.

All substantial and invalid test informational collections are to be set up for each experiment and a test information archive is readied. Test information is additionally created in light of some calculation and instruments [14]. Experiment readiness [11] has different advances which begin with Test case age [12], Test case determination [16], Evaluation, and Test case prioritization [5][18]. There are different calculations which are utilized to create and upgrade test cases [19][20].

Swain et.al proposed a procedure to create test cases utilizing comparing succession charts and furthermore indicates the limitations over the characterized antiques. In the meantime experiment age systems are useful for recognizing synchronization and reliance of utilization cases and messages, question communication and task flaws [12]. Test condition arrangement is a standout amongst the most essential stages which are generally arranged by particular group dealing with situations. After fruition of coding part, the code is checked by setup administration apparatus and afterward test assemble is readied where analyzers need to begin test execution[21].

3.3 Test Execution Phase

In this stage analyzers execute programming according to test cases. Wherever real and expected outcomes don't coordinate then analyzer open bugs and relegate the same to designers. Bug logging and following [13] takes after entire life cycle of bug. There is as of now a great deal of work which has been done in past that spot-

lights on principle ventures to be taken to report legitimate blame. The standard reports can be examined on week by week/everyday schedule alongside the ventures advance on venture conveyance, acknowledgment and endorsements are checked to break down pilot project[23].

3.4 Test Closure

Test Closure is an imperative stage which incorporates all test reports guaranteeing that all framework, coordination, client acknowledgment testing passed and choice is taken whether all necessities are tried and there is no basic bug pending to be settled OR checked. An audit of all test antiquities is finished by Manager[22]. When all antiques are investigated and endorsed then programming discharge is finished. Additionally main driver investigation is being done to conceptualize on what went well, what did not go well and zones of change. There are different main driver examination instruments and techniques accessible on which a loads of research has been done in past.

4 TYPES OF SOFTWARE TESTING

Testing is associated with each phase of programming life cycle, however the testing done at each level of programming advancement is distinctive in nature and has diverse targets.

Unit Testing is done at the most reduced level. It tests the essential unit of programming, which is the littlest testable bit of programming, and is regularly called "unit", "module", or "part" reciprocally.

Integration Testing is performed when at least two tried units are consolidated into a bigger structure. The test is frequently done on both the interfaces between the segments and the bigger structure being built, if its quality property can't be surveyed from its segments.

System Testing has a tendency to assert the conclusion to-end nature of the whole framework. Framework test is regularly in light of the useful/necessity detail of the framework. Non-utilitarian quality properties, for example, dependability, security, and practicality, are additionally checked[14].

Acceptance Testing is done when the finished framework is given over from the engineers to the clients or clients. The reason for acknowledgment testing is preferably to give certainty that the framework is working than to discover blunders.

White box and Black Box Testing Black box testing is performed to guarantee yield of utilization is as right for every single different sort of positive and negative sources of info. There are different sorts of Black box testing writes like Equivalence Class parceling, Boundary esteem examination, blunder speculating and so forth. White box manages inside working of code to guarantee there is no excess code written in programming. This includes testing of line of code, program, stream, rationale, circle, structure, capacities, class correspondence testing and other inside testing of program.[15]

5 CLASSIFICATION OF TEST TECHNIQUES

In this paper, the most important test techniques are shortly described, as it is shown in Figure 1.

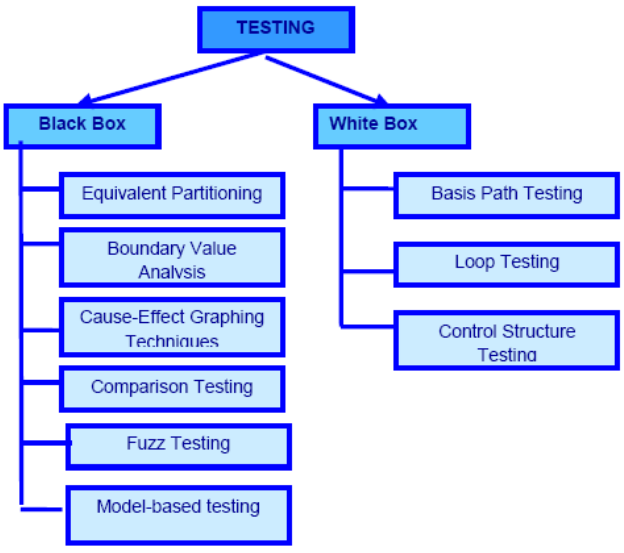

Figure 1: General Classification of Test Techniques

WHITE BOX TESTING

White box testing is very successful in recognizing and settling issues, since bugs can regularly be found before they cause trouble.[5] White box testing is the way toward giving the contribution to the framework and checking how the framework forms that contribution to create the required output.White box testing is additionally called white box investigation, clear box testing or clear box analysis.[5] White box testing is appropriate at coordination, unit and framework levels of the product testing process.[3] White box testing is considered as a security testing strategy that can be utilized to approve whether code usage takes after proposed configuration, to approve actualized security usefulness, and to reveal exploitable vulnerabilities.

Some Different types of white box testing techniques are as follows:-

1. Basis Path Testing
2. Loop Testing
3. Control Structure Testing

Advantages Of White Box Testing:-

1. All independent paths in a module will be exercised at least once.
2. All logical decisions will be exercised.
3. All loops at their boundaries will be executed.
4. Internal data structures will be exercised to maintain their validity.
5. Errors in hidden codes are revealed.
6. Approximate the partitioning done by execution equivalence.
7. Developer carefully gives reason about implementation.

Disadvantages Of White Box Testing:-

1. Missed out the cases overlooked in the code.

2. As information of code and interior structure is an essential, a talented analyzer is expected to do this sort of testing, which expands the cost.

3. And it is about difficult to investigate all of code to discover shrouded blunders, which may make issues, bringing about disappointment of the application.

2) BLACK BOX TESTING Black box testing will be trying programming in view of yield prerequisites and with no information of the interior structure or coding in the program.[5] Basically Black box testing is a vital piece of Correctness testing yet its thoughts are not constrained to accuracy testing as it were. The objective is to test how well the segment complies with the distributed necessity for the segment. Discovery testing have next to zero respect to the inner sensible structure of the framework, it just inspects the essential part of the framework. It ensures that info is appropriately acknowledged and yield is effectively created. [3]

Some Different types of Black box testing techniques are as follows:-

1. Equivalent Partitioning

2. Boundary value Analysis

3. Cause-Effect Graphing Techniques ı

4. tem Comparison Testing

5. Fuzz Testing

6. Model-based testing

Advantages Of Black Box Testing:-

1. The quantity of experiments are lessened to accomplish sensible testing

2. The experiments can indicate nearness or nonattendance of classes of mistakes.

3. Black box analyzer has no "bond" with the code.

4. Programmer and analyzer both are autonomous of each other.

5. More compelling on bigger units of code than clear box testing.

Disadvantages Of Black Box Testing:-

1. Test cases are difficult to outline without clear determinations.
2. Only little quantities of conceivable info can really be tried.
3. Some parts of the back end are not tried by any stretch of the imagination.
4. Chances of having unidentified ways amid this testing
5. Chances of having reiteration of tests that are as of now done by developer

6 TRENDS FOR FUTURE

In this paper, we have given a thorough on various kinds of test information age systems. The paper gives a general thought in the field of mechanized test information age. Our work may help the peruser to choose the territory and concentrate the ideas of related region. Subsequent to concentrate there related territory of references the peruser may pick one without bounds challenge zones expressed in segment 5 for research and they may chase here. We have recorded the heading of research in a compact manner[26].

6.1 Direction of Research

1. Improvement of productive develop solver.
2. Observational examination for discovering heuristic for various kinds of programming build to decrease the quantity of cycle required for test information age.
3. Circle Bounds and Infeasible way recognition for WCET Analysis
4. Change of adaptability of test information age extraordinarily organizing programming.

5. Test information age without way selector. Shirking of infeasible way amid test information age process.
6. Scope in view of need of code portion. Prioritization of code concerning scope to lessen be taken a toll.
7. Change of test information age for programming develops having dynamic information structure.
8. Test information age for programs having variable length cluster and circles with various measurements.
9. Test information age for test viability and code scope Analysis.
10. A Comparative investigation of various test information age execution procedures to decide best technique for programs.
11. Testing object situated programming utilizing UML demonstrating.
12. Test information age for recursive projects and strategies/capacities.
13. GUI testing.
14. Agile testing.

7 PERFORMANCE EVALUATION

7.1 Test Scenario

Keeping in mind the end goal to ensure the exactness and convenience of the VC data transmission, it is important to break down and follow up on the information in under a moment. Information are sent to the Cloud for chronicled examination, BD investigation, and long haul stockpiling with less inertness. For instance, every one of the thousands or a huge number of FNs may send occasional synopses of the information network to the Cloud for authentic investigation and capacity.

In the test completed, the FNs are thought to be set over a spatially constrained Intranet. We consider a physical topology with four heterogeneous FNs. Keeping in mind the end goal to

show the CPU utilizations of the FNs, we allude to the Intel R-CoreTM 2 CPU Q6700 with 2.67 GHz recurrence rate and 4 GB of RAM memory. To affirm the accuracy of the proposed foundation in Figure 1, we require an assessment to investigate a few asset administration and planning methods, for example, administrator, application, errand arrangement and combination. We assess some asset administration approaches appropriate to the Fog condition concerning their effect on dormancy, arrange use, and vitality utilization. In this structure, the VCs distribute information to SVF; applications are keep running on Fog gadgets (FN) to process information originating from VCs. Without a doubt, it is essential to test the arrangement with a certifiable info workload and contrast it and the comparing arrangements of the current FDC-based systems. Along these lines, in the following subsection, we will proceed onward to this issue and address it with a few reproduced tried cases[17][18].

7.2 Test Result

In this subsection, we expect to test our scheduler on 100 approaching information streams (T = 100) for a Fast Ethernet correspondence channel and ascertain the on-line momentary general vitality utilization and vitality investment funds that are appeared in Figure 4. From this figure, we reason that the aggregate vitality cost changes after some time; the figure even demonstrates the vitality sparing related to time.

8 CONCLUSION

The intent of this paper was to research on various phases of software testing life cycle and different types of testing. After reviewing various phases of software life cycle it is found that there are main 4 phases in testing life cycle that could be categorized as Analysis, Planning and Preparation, Execution and Closure. A generic software testing life cycle- APEC is proposed in this paper. Also most recent failures are studied, which happened due to lack of performance and security testing. A lot of time is spent on Functional testing and there is rarely any software which got crashed due to lack of functional testing in recent past. So this paper proposed a

new right mix of testing which should include some performance and security testing checks in addition to functionality testing for better quality of software. As there is always a scope so Further to this paper a research and study can be done on the software testing to propose a generic testing framework and techniques to support functional, performance and security testing for object oriented development framework and other platforms using some algorithm(s) with/ without use of tools in minimum amount of time.

References

[1] Karagiannis, G.; Altintas, O.; Ekici, E.; Heijenk, G.; Jarupan, B.; Lin, K.; Weil, T. Vehicular networking: A survey and tutorial on requirements, architectures, challenges, standards and solutions. IEEE Commun. Surv. Tutor. 2011, 13, 584616.

[2] Naranjo, P.G.V.; Pooranian, Z.; Shojafar, M.; Conti, M.; Buyya, R. FOCAN: A Fog-supported Smart City Network Architecture for Management of Applications in the Internet of Everything Environments. arXiv 2017, arXiv:1710.01801.

[3] Bonomi, F.; Milito, R.; Zhu, J.; Addepalli, S. Fog computing and its role in the internet of things. In Proceedings of the First Edition of the MCCWorkshop on Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; pp. 1316.

[4] Mahmud, R.; Kotagiri, R.; Buyya, R. Fog computing: A taxonomy, survey and future directions. In Internet of Everything; Springer: Berlin, Germany, 2018; pp. 103130.

[5] Tajiki, M.M.; Salsano, S.; Shojafar, M.; Chiaraviglio, L.; Akbari, B. Joint Energy Efficient and QoS-aware Path Allocation and VNF Placement for Service Function Chaining. arXiv 2017, arXiv:1710.02611.

[6] Shojafar, M.; Cordeschi, N.; Baccarelli, E. Energy-efficient adaptive resource management for real-time vehicular cloud services. IEEE Trans. Cloud Comput. 2016, 114, doi:10.1109/TCC.2016.2551747.

[7] Bsching, F.; Schildt, S.;Wolf, L. DroidCluster: Towards Smartphone Cluster ComputingThe Streets are Paved with Potential Computer Clusters. In Proceedings of the 2012 32nd International Conference on Distributed Computing Systems Workshops, Macau, China, 1821 June 2012; pp. 114117.

[8] Bogdan Korel, Ali M. Al-yami, Department of Computer Science, Illinois Institute of Technology, Chicago, IL 60616s, Generating Fast Code From Concurrent Program Dependent Graphs, Proceedings of ICSE-18, 1996 IEEE.

[9] C. Cadar and D. Engler, Execution Generated Test Cases: How to Make systems Code Crash Itself, Technical Report, Computer Systems Laboratory Standford University, Standford CA-94305, U.S.A. 2005.

[10] M. Gittens, K. Romanufa, D. Godwin, J. Racicot, All Code Coverage is not created equal: A case study in prioritized code coverage, Technical Report, IBM Toronto Laboratory, 2006.

[11] Mary Jean Harrold Gregg Rothermel Alex Orsa, Representation and analysis of software.

[12] Rullian Zhao and Qing Li., Automatic test generation for dynamic data structure , Technical report, Beijing University of Chemical Technology, 2006.

[13] Kiran Lakhotia, Phil McMinn, and Mark Harman, Automated Test Data Generation for Coverage: Havent We Solved This Problem Yet?, 2009.

[14] Clay E. Williams Center for Software Engineering IBM T. J. Watson Research Center, 2009

[15] Bogdan Korel, Ali M. Al-Yami, Automated Regression Test Generation", ISSTA 98 Clearwater Beach Florida USA , 1998.

[16] B.Beizer .1995.Software Testing Techniques.2006.Van NostrandReinhold,New York.1990.ISBN.0-442-20672-0.(31.Oct.2006).

[17] A,Bertolino.2001.Chapter 5: Software Testing . IEEE SWEBOK trial version 1.00.IEEE(May 2001).

[18] Khan,Mohd.Khan,Farmeena.2012.A Comparative Study of White Box, Black Box and Grey Box Testing Techniques.2012. International Journal of Advanced Computer Science and Applications(IJACSA). Vol. 3.No.6.(2012).

[19] Tarika,Bindia. Computer Programmer CSE,GNDEC, Ludhiana,Punjab-India.IJRITCC.2,1 .68-72.(2321-8169).

[20] B,Swarnendu.R,Mall.CSeDeptt,IIT Kgp. 2011. Regression Test Selection Techniques, A Survey-Informatica 35 .2011.

[21] Swain,S.k.Mohapatra,D.P.Mall,R.2010.Terst Case Generation Based on Use Case and Sequence Diagram.International journal of Software Engineering (IJSE).3, 2.(July 2010),(289-321).

[22] Thakre, Sheetal. Chavan,savita.Chavan,P.M.] 2012. Software Testing Strategies and Techniques.International Journal of Emerging Technology and Advanced Engineering .Website: www.ijetae.com .2, 4.(April 2012), (2250-2459).

[23] An Approach to Cost Effective Regression Testing in Black-Box Testing Environment - IJCSI International Journal of Computer Science Issues. 8, 3, 1(May 2011),(1694-0814).

[24] Chauhan,Kumar,Vinod.2014.Smoke Testing- International Journal of Scientific and Research Publications4,2 (February 2014),(2250-3153).

[25] Gupta,Varuna.Sen,Saxena,Vivek.2013.Software Testing: Smoke and Sanity- International Journal of Engineering Research & Technology (IJERT).2,10(October 2013) (2278-0181).

[26] Liskin,olga.Hermann,christoph.Knauss,Eric.Kurpic, Thomas .Rumpe,Bernhard.Schneida,Kurt.2012.Supporting Acceptance Testing in Distributed Software Projects with Integrated Feedback Systems: Experiences and Requirements. IEEE Seventh International Conference on Global Software Engineering.(2012).