DOI: 10.7763/IPEDR. 2014. V75. 6

Testing and Evaluation : A case study Approach

Hema Gaikwad, Samaya Pillai

Symbiosis International University

Abstract. In the Binary world software development is the most important functionality; Software Testing is one of the phases of software development life cycle. Customer or End users perception is that after testing no errors or bugs will exist in the software; or technically we can say that "Exhaustive Testing is possible". But at actual this is not true. We cannot rectify each and every error, some errors will exist and the customer will never be satisfied 100%. But using quality tools and techniques we can minimize the errors optimally. this This paper explores various software testing principles and focuses on one of the software testing principle - "EXHAUSTIC TESTING IS IMPOSSIBLE". The research study analyses deployed software application (Beta tested software), which still have different types of errors and application of quality tools and techniques which can reduce them

Keywords: Exhaustive testing, Beta testing, Defect, Error

1. Introduction

What is software testing? Software testing term consists of two different and independent terms software and Testing. Software is a set of instructions or programs and Testing is to check those instructions or programs; so software testing is a process by which we can identify and minimize the errors from the application software. It is stated that Software testing is the process of exercising a product to verify that it satisfies specified requirements or to identify differences between expected and actual results [2]. It is mentioned that A process of executing a program with the objective of finding an errors with the help of finite no of test cases is known as testing, and the purpose is to test the application by using minimum time and minimum efforts [3].

What is Process? The process is set of activities related to each other and perform a specific task.

What is Product? Output of the process is known as Product.

What is Test-Case? A set of input values, execution preconditions, expected results and execution post conditions developed for a particular objective or test condition, such as to exercise a particular program path to verify compliance with a specific requirement. It is stated that we can categories the test cases-Logical test cases and concrete test cases. Logical test cases have to be defined first; after the logical test cases can be translated into concrete physical test cases meaning that actual input are chosen [8].

What is Defect? When we write the Test cases we mention both the expected result and actual result. If both the results are same then there is no defect but when there is difference between actual result and expected result that means there is a defect.

It is mentioned that an incorrect step, process, or data definition in a computer program is called a defect. Software defects are unique compared to physical defects in a product. They are harder to assess, and more cunning. Software does not deteriorate like physical products. The deterioration of software is a product of side effects from changes in the product in order to assess defects or changed requirements. This implies the total defect count may increase after release during maintenance. There is a risk of introducing new defects when changing the code. Software defects can be injected during any phase of the software development life cycle. The following as sources of defects: requirement errors, design defects, coding defects, documentation1a defects and incorrect corrections A defect is injected when an employee does a mistake in the work performed which creates a defect. A failure in the software might manifest itself and be discovered in either testing or inspections. However, not all defects are discovered before delivery. Such defects are

1 Corresponding author. Tel.: + 9503087775; 9890140505; fax: +(25675603).
E-mail address: hemagaikwad2005@gmail.com

latent defects and can be harder to locate, and advanced users are more likely to discover these than normal users [6].

What is Error? A human action that produces an incorrect result is known as error.

2. Importance of Software Testing

In software engineering three P's are important PROBLEM, PROCESS and PRODUCT. Problem and Product are the input and output of the PROCESS respectively. Process is a set of activities together with an orderly relationship between activities to produce the desired output. Software development life cycle (SDLC) is one of the processes and its activities are Requirement Analysis, Feasibility study, Planning, Analysis, Design, Coding, Testing, Quality, Implementation and Post Implementation [5]. It mentioned that The process used to create a software product from its initial conception to its public release is known as system development life cycle model [7]. Software testing activity plays a very important role. We can perform testing only when the code is ready. This activity has its own independent life cycle and consists of 5 phases. It starts from Requirement Analysis, Test plan, Test case development, Environment setup, Test case execution and Test close. This activity is very essential because we cannot deploy the software directly because there may be errors so testing helps to identify the errors and check its severity. Software testing has various principles and the details are as follows—

3. Principles of software testing

Principle2: Exhaustive testing is impossible: It is also known as complete testing and its nature is very exhaustive. In this type of testing we try to check the output given by the software by entering all combinations of inputs and preconditions. Each element of code is verified under this process. Exhaustive testing is usually done when the programs and the scope of project is small. For bigger projects exhaustive testing is impossible & impractical and is not used.

Exhaustive Testing – Perceptions: Neutral perception (Layman's)--When we develop the software and after development when we deploy the software to the end user, the user's expectation is software should be complete in all senses. It means that it should be functionally correct whatever input we give output should be display accordingly. Finally we can say that he/she wants the complete and correct product.

Tester's Perception—Software testing team members are the project stakeholders. Their perception is "COMPLETE TESTING (EXHAUSTIVE TESTING) IS IMPOSSIBEL". Team takes more efforts and perform various types of testing on the application, examples of testing's are Functional (Black Box), Structural (White Box), Unit, Integration, System, User acceptance, Alpha, Beta, Regression, Load and Stress. The details are as follows-

What is Functional Testing? It is also known as Black Box testing. We check the functionality of the application; we give the input and check the output correspondingly.

What is Structural Testing? It is also known as White Box testing. We check the complete code, control flow and all control structures used in that program.

What is Unit testing? When we start with a project first we decompose it into smaller modules or units, this process is known as decomposition or modularization. After decomposition we get different modules and perform functional and structural testing on them. It is stated that unit testing is the micro testing, and it is performed on the code or program. It is performed by software developer not by tester [4].

What is Integration Testing? When all the modules are unit tested then we start integration. Integration means tie each units according to the flow; parallelly we also perform functional and structural testing. After integration if the result is correct then we continue the same process until the complete integration is complete. It is mentioned that Integration testing is building a system from its components. Top-down integration is develop the system and move downwards i.e. towards components. Bottom-up integration is start from integrating the components and then it becomes a complete system [4].

What is System Testing? After integration it becomes a complete system then we check the functional requirement (WHAT REQUIREMENTS) as well as the non functional requirements (HOW REQUIREMENTS) such as Performance, Reliability, and user-friendliness.

What is User Acceptance Testing? When the system is tested then we perform validation testing such as Alpha and Beta testing. Both validation testings are performed by end users. When the user tests the application on the developer's site with dummy data it is alpha testing whereas when user tests the application on its own site with actual data or real data it is beta testing. When the project or Application is beta tested it is assume that project is ready and correct. Now we can launch it in a market.

What is Regression Testing? If your project is ready for deployment or it is already deployed and now customer wants some changes then developer perform that changes and again tester will test the change area as well as its affected area also. This is regression testing. It is mentioned that An important regression testing strategy is to place a higher priority on testing the older capabilities of the product than on testing the new capabilities [2].

What is Load and Stress Testing? For every application we check the load and stress. Load means how many people use the application at a time and the Stress mean how many client access or retrievals are done for the information from the server. It is mentioned that when we test the application under heavy loads. For example When we test the website under a heavy load the system response time degraded or fails [4].Although the tester checks functionality, user interface, Navigation, Database retrievals as well as the error handling etc but the word “BUT” still exists. That's why the tester perception is “EXHAUSTIVE TESTING IS NOT POSSIBLE”.

4. Methodology

This Paper has a case-study approach. The research study analyses the Deployed software application (Beta tested software) which have still different types of errors.

5. Data: Case Study

Exhaustive Testing –in different context : a) For Small project Exhaustive testing is possible.

b) For large and complex project exhaustive testing is not possible. Software testing is context dependent. It means that it vary from domain to domain. We consider two scenario's. First scenario is for Desktop applications and the other is for web based applications.

Desktop Application: Consider the example of simple calculator with only three text boxes, two for variable and one for result. If A,B are variables and C is constant.

Small scope: Suppose we increase the scope that is it accept the integer and float values with four primary operations like addition, multiplication ,division and subtraction.

Integer Values : 12 combinations are required to check the various conditions.

Float Values: For addition we require - 1. A=B ; 2.A>B; 3. A<B; 4. A is not equal to B

For subtraction we require - 1. A=B; 2.A>B; 3. A<B; 4. A is not equal to B

For multiplication we require - 1. A=B; 2.A>B; 3. A<B; 4. A is not equal to B

For division we require: 1. A=B; 2.A>B; 3. A<B; 4. A is not equal to B

That means if we include the float values again you have to check the 12 more combinations. That means you have to check total 24 values. Exhaustive testing is possible. Performance and satisfaction rate is high.

Medium scope: If we increase the number of operators for example square root and percentage etc. The no. of combinations will increases but it will be in the limit. Exhaustive testing is possible. Performance and satisfaction rate is high.

Large Scope: Consider the scientific calculator even though it is large and complex compare to the very basic calculator. Exhaustive testing is possible. Performance and satisfaction rate is high.

Conclusion is If the project is very small and simple we say that Exhaustive testing is possible.

Web based Application: When we run any application through various browser's (IE, Fire Fox and Google chrome) or when we access through web it is called web based application. Client server architecture is used for Web based Application. Web based application can be accessed centrally or distributively.

For case study IRCTC (Indian Railway Catering & Tourism Corporation) web site was considered.

IRCTC is an e-ticketing Indian website where we book railway tickets for travel ling all across the country. It makes it easy for users to book tickets with certain beneficiaries. Users book the tickets and receive an acknowledgement in return which may be in the form of an email or an SMS. Tickets of all the types and quotas can be booked here efficiently. Users can make use of all the privileges available on the website. At the early stage the user is asked to sign up if already not a registered member, else sign in with the user id and password provided. After signing in the user can search to the desired train and its respective timings and all the associated details of the same. Finally the user can book the ticket by making the on line payment and authorizing itself through a secure payment gateway.

There are various modules such as Ticket Booking, Ticket Cancelling, Train search criteria, Passenger History and Payment details etc. IRCTC has various features as well as drawbacks. Some of the drawbacks are as follows-

- On the festive seasons when the load increases the page navigation rate become low as well as it increase the stress on the server.
- Slow speed
- As the users click on text boxes to fill in the details or select any check boxes or drop downs, a pop up of advertisements appear on the screen ,making it less user friendly.
- There is a major problem in current booking system and payment gateway mode is session time out.
- When the user opts to make the payment ,if a delay happens the whole session gets expired and the user has to restart the whole session in order to proceed with the payment.

Sometimes, after the payment the confirmation mail or message delivery fails resulting into convenience to the user.

6. Quality Assurance and Quality Control

Quality Assurance: :As we know any process whether it is IT (Information Technology) or Non IT process, consist of number of sub processes. Software development is Information technology process. It consist of various sub processes such as Requirement collection and segregation process, Planning process, Designing process, Development process and Deployment process. The Venn diagram as an artifact can be used for representing the process and sub-process.

Quality Assurance is process oriented approach that means QA team is associated with main process as well as the sub processes. QA team is responsible for checking the concern process as well as the intermediate deliverables. If we consider the sub processes of software development such as Requirement collection and segregation process the intermediate deliverable is Software requirement specification, The High level and low level plan documents are intermediate deliverables for Planning process, Design prototype is the deliverable for Designing process, Code is the deliverable for Development process and Deployment process. Final product we get after deployment. The QA team is more focused on defect prevention and quality or it is on prevention side. It stresses more on designing capable processes.

Ishikawa diagram or Fish bone diagram or The root cause analysis is one type of quality metric. We can measure the quality by qualitatively and quantitatively. The above metric comes under qualitative measurement. We can draw the Fish bone diagram for both type of processes such as IT and Non IT. The diagram has basically two sections left and right. Left hand side we write all the causes and their effects and in Right side we write the actual problem. We can categorize the causes under different heads they are as follows- 1. MEN 2.MACHINES 3. MATERIALS 4. METHODS 5.MEASUREMENT 6.MOTHER'S NATURE/ ENVIRONMENT.

They are also known as 6M's of Fish bone diagram. If there are problems in any process so Fish bone diagram is the best quality metric. The format of the Fish bone diagram is as follows--

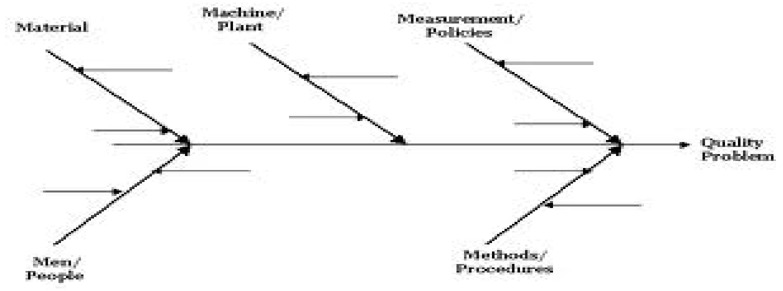

Figure 1. Root Cause Analysis

Quality Control: QC is a product oriented approach and focuses on defect identification. It is on correction side. QC stress on inspection and testing the activities. Under quality control there are so many testing techniques and testing types.

There are two types of organizations available in the market 'q' organization and ‘Q’ organization. The ‘q’ organizations are those which are less quality conscious, and ‘ Q' organization lay more stress on prevention activities. These are more quality conscious organizations. Most of the organizations are ’Q’ organization.

7. Application of QC to medium and large projects

With reference to our web based project (our case) we suggest that if the deployable product (Beta version) still consist of multiple errors so the good practice is to find out all the causes find out their effects. If we remove or minimize the problems then quality and customer satisfaction will automatically increase.

8. Findings

When we launch a product in the market it is mandatory that we have to take the continuous feedback and follow up. If the feedback is positive then there is no problem but if the feedback is negative then the application requires changes. Improvements are requiring in case of negative feedback.

9. Conclusion

This all things proves that The Beta tested software (Completed and Corrected Software) also show some errors. This means the sentence the “EXHAUSTIVE TESTING IS IMPOSSIBLE” is true but we can reduce the problems or defects by using the Fish bone diagram considerably.

10. References

[1] Dorothy Graham, Erik van Veenendaal ,Isabel Evans and Rex Black. Foundations of software testing. US,2006.

[2] Omprakash Deshmukh, Mandakini Kaushik “A Overview of Software Verification &Validation and Selection Process” International Journal of Computer Trends and Technology volume4Issue2- 2013

[3] Jovanović, Irena, Software Testing Methods and Techniques, May 26, 2008 DIV Inzenjering, d.o.o., Belgrade

[4] Shivkumar Hasmukhrai Trivedi, Software Testing Techniques, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 10, October 2012

[5] Roger Pressman, Software engineering,7th international edition, 2010

[6] Jan Maximilian Winther Kristiansen, Software Defect Analysis, 15. January 2010

[7] Ron Patton, Software Testing,UK,2005

[8] Elfriede Dustin,Effective software testing,UK,2006