Role of Testing in Software Development Life Cycle

Article *in* International Journal of Computer Sciences and Engineering · May 2019

DOI: 10.26438/ijcse/v7i5.886889

1 author:

Nirali Honest
Charotar University of Science and Technology
16 PUBLICATIONS **68** CITATIONS

SEE PROFILE

International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-5, May 2019 E-ISSN: 2347-2693

Role of Testing in Software Development Life Cycle

Nirali Honest

Smt. Chandaben Mohanbhai Patel Institute of Computer Applications,CHARUSAT,Changa

*Corresponding Author: niralihonest.mca@charusat.ac.in,

DOI: https://doi.org/10.26438/ijcse/v7i5.886889 | Available online at: www.ijcseonline.org

Accepted: 08/May/2019, Published: 31/May/2019

***Abstract*—** People search for quality in every artifact they come across. The concept of quality has also entered in the area of software development where it becomes crucial to thoroughly check the software system at different levels of testing. Now-a-days the competition is increased highly and the frequency of changes in platforms and business requirements are also very high, so for a software to be stable and in use for long run, requires to support and update based on the current requirements. Software testing is one of the umbrella activities performed at any organization to provide value and quality, to ensure the longitivity of software product in the market. This paper covers the concept of testing, its role in assuring quality, test cases, levels of testing, methods of testing and test planning, executing and monitoring. The papers emphasize on the use and impact of test driven environment with concept of story board based implementation.

Keywords* — *Software testing, Software quality, Test Driven Environment.

I. INTRODUCTION

Introduction

The concept of quality did not originate with software systems [8]. It started with as an impact of Global competition, Outsourcing, growing customer outlooks, have brought the concept of quality to the considerably highlighted. Initially the quality movement started in Japan with the concept of Statistical Quality Control (SQC) methods. Statistical quality control is a discipline based on measurements and statistics. Decisions and Plans are made and based on the gathering and calculation of the actual data in the form of different metrics, the past experience or intuition is not considered as the major way of doing the work. A working model based on the concept of forming a Strategy, Prepare and Organize to work as per the strategy, Assess the work and Manage changes was adopted where initially a plan or strategy was established to set the objective and process to deliver the results, then implement the plan and measure its performance, and then assess the measurements and report the results to decision makers, and Apply changes if required to improve the process. Basically quality concept started with one unit in the organization, but then it moved to the entire organization. This caused the move from SQC to TQC (Total Quality Control) in Japan. With the acceptance of quality standards to be enforced in the entire organization various quality related activities were introduced in the organization like audits, quality circle promotion of quality management principles and quality control. One of the innovative TQC methodologies developed in Japan is referred to as cause-and-effect model. This was also called as the Ishikawa model and it is used to understand the causes that has effect on the quality of the product. The Software Quality is defined as "An effective software process applied in a manner that creates a useful product that provides measurable value for those who produce it and those who use it". Effective testing is very important to perform which include minimum test cases with maximum cover [1]. Testing is a costly activity if not carried out optimally, and its aim is to find defects from systems with critical and catastrophic events [2][3][4]. There are two important terms used in testing which include 'Validation' and 'Verification'. Verification activity helps us in assessing a software system by determining whether the product of a given development phase satisfies the requirements established before the start of that phase. One may note that a product can be an intermediate product, such as requirement specification, design specification, code, user manual, or even the final product. Activities that check the correctness of a development phase are called verification activities. Validation Activities help us in confirming that a product meets its intended use. Validation activities aim at confirming that a product meets its customer's expectations. In other words, validation activities focus on the final product, which is extensively tested from the customer point of view. Validation establishes whether the product meets overall expectations of the users. Validation is done at the

end of the development process and takes place after verifications are completed [5]. The testing activities are challenging activities and a proper roadmap is to be decided before proceeding to testing [9][10].

II. ROLE OF TESTING IN SOFTWARE QUALITY

Testing plays a significant role in accomplishing and evaluating the quality of a software product [13]. Software quality assessment can be divided into two broad categories, namely, static and dynamic analysis. Static Analysis means, it is based on the examination of a number of documents, namely requirements documents, software models, design documents, and source code, etc. Basic methods used in static analysis includes code review, inspection, walk-through, algorithm analysis, and proof of correctness. It does not involve actual execution of the code under development. Dynamic Analysis: Dynamic analysis of a software system involves actual program execution in order to expose possible program failures. The behavioral and performance properties of the program are also observed. Programs are executed with both typical and carefully chosen input values. Often, the input set of a program can be impractically large. The major objectives of testing include, a. The system will work properly, b. The system will not work properly, c. Reduce the risk of system breakdown, and d. Reduce the cost of testing.

The first objective covers the test cases to check the system will work properly with all specified operations this test intend to check the basic flow of use case model and it covers the positive testing scenario, the second objective covers the test cases to make the system fail, so check all the alternative flows specified during the use case model and it covers the negative testing scenario, the third objective covers the test cases to fail or crash the system explicitly, to know the limit and boundaries of system, so likewise the maximum system capacity in terms of processing speed, memory allocations, interrupt handling, etc. can be known, the fourth objective focuses on designing of minimum test cases to cover the maximum system testing, it is near to impossible to test all the combination of test cases ,so the challenge of testing is to cover entire system testing with minimum number of test cases and utilize the resources optimally with reducing the cost associated with it. The test case generation is one of the most important task in testing. [15]. A test case is a simple pair of <input, expected outcome>. In stateless systems: output completely depends on the input, example includes to compute the cube of numbers, user simply needs to provide input to the system. In state-oriented systems: output of one sequence act as an input to another sequence and so on, example includes a telephone switching system and ATM machine, etc. The test cases can be formed from different sources of information as a software development process produces a large number of artifacts, such as requirements specification, design document, and source code. In order to generate effective tests at a lower cost, test designers study and analyze the various sources of information like, Requirements and functional specifications, Source code, Input and output domains, Operational profile, and a fault model which predict the incorrectness in the system. From this model, a designer can predict the consequences of particular faults. In order to test a program, a test engineer must perform a sequence of testing activities that include, identify an objective to be tested, select inputs, compute the expected outcome, set up the execution environment of the program, Execute the program, Analyze the test result.

III. LEVELS OF TESTING

Testing is performed at different levels involving the complete system or parts of it throughout the life cycle of a software product.

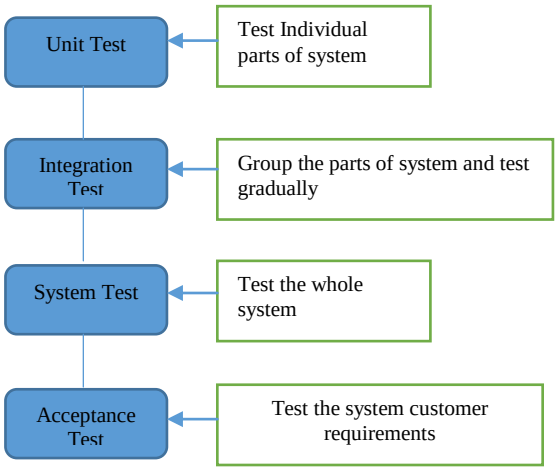

Figure 1: Levels of Testing

In Unit level testing the individual units/components are tested in a standalone manner. The purpose is to validate that each unit of the software performs as designed without any dependency. Integration Testing is a level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. System testing is a level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's agreement with the specified requirements and Acceptance testing is a level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.

IV. METHODS OF TESTING

Two broad concepts in testing, for test design, are white-box and black-box testing [6][7]. White-box testing techniques are also called structural testing techniques. As the name implies this testing methodology provides the insight into the application and create test cases. The tester has the source code structure and other details of application available, so based on it the test cases can be prepared. It enables to see what is happening inside the application. For example, consider a system Bank ATM machine, here the code behind the system, would be to provide operations for pin verification, listing menu items, check balance, withdraw cash, link aadhar card, etc. As part of white box testing the tester focus on working of each module correctly, before it is tested using the user interface. Here the main source of input for test cases is the underlying structure of the source code. This method of testing can be applied at unit level and integration level testing where the code is developed and tested in parallel. Black-box testing techniques are called functional testing techniques. As the name implies the tester is not provided with an insight of the application that is being tested [11][12]. The tester understands the requirements and test them using the user interface of the system. The tester is not aware of the code behind the interface being tested. This testing methodology looks at what are the available inputs for an application and what the expected outputs are that should result from each input. For example, consider a system Bank ATM machine, the tester understands the possible operations supported by the ATM, prepares the set of inputs and expected outcomes to test each operation, at the time of testing the tester has access of user interface only, the source code is not made available to the tester. Here the main source of input for test cases is number of inputs and expected outputs to test the user interface. This method of testing can be applied at system level testing where all the interfaces are prepared and tested for each functionality. The main difference between black-box and white-box testing is the areas on which they choose to focus. In simplest terms, black-box testing is focused on results. If an action is taken and it produces the desired result, then the process that was actually used to achieve that outcome is irrelevant. White-box testing, on the other hand, is concerned with the details. It focuses on the internal workings of a system and only when all paths have been tested and the sum of an application's parts can be shown to be contributing to the whole is testing complete.

V. TEST PREPARATION AND POLICY

The purpose of system test preparation includes setting the test planning, to get ready and organized for test execution [14]. A test plan provides a framework, scope, details of resource needed, effort required, schedule of activities, and a budget. In earlier days the traditional testing concept was followed as shown in figure 2, where testing is done at the end, this has direct impact on the time and cost of the software which is not feasible, it is interesting to note that a new test-centric approach to system development is gradually emerging. This approach is called test-driven development (TDD). In test-driven development, programmers design and implement test cases before the production code is written as shown in figure 3. This approach is a key practice in modern agile software development processes such as eXtreme Programming (XP). The XP is a software development methodology which is used to improve software quality and responsiveness to changing customer requirements.

Figure 2 : Traditional concept of testing.

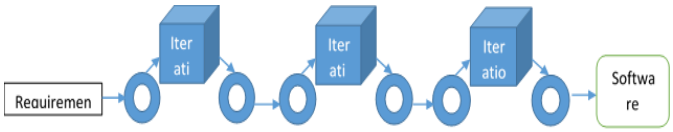

Figure 3 : Test Driven Development in testing.

Extreme programming" (XP) champions the use of tests as a development tool, and it supports the "Test First" functionality. Here the requirement are taken up as stories and they are written on the story board, one story at a time is written to meet the functionalities and the code to test the story is also written, once the test is passed and meet all the requirements from a given story, the next story from the story board is taken, and it continues up to all the stories are written and tested as shown in figure 4.

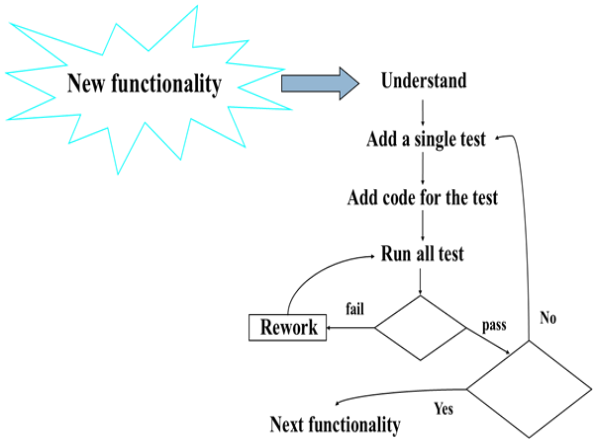

Figure 4: Testing the functionalities in a test driven environment.

The main features of agile software development practices include Incremental development means to divide the software life-cycle development into "multi-waterfall" cycle and with every cycle the testing is contributing to the changes that need to be incorporated, the Unit testing is done by the programmers, and acceptance testing done by customers, on more frequent basis regression testing is done to make sure that the changes made in the code, database, etc. to support the new requirements do not disturb or cause problems in already existing code. It also focuses on writing test code for one test case at a time, before the final code is written. After the test planning and design monitoring and measurement are two key principles in software testing. Different software matrices are created to store and analyze the results. Software Testing Matrix is a quantitative measure of the degree to which a system, component, or process possesses a given attribute. Test execution metrics can be broadly categorized into two classes as Metrics for monitoring test execution and Metrics for monitoring defects. Quantitative evaluation is carried out through measurement.

Testing is a distributed activity conducted at different levels throughout the life cycle of a software. In testing, team is organized and managed Unit-level tests are generally developed and executed by the programmers. System integration testing is performed by the system integration test engineers. A team for performing system-level testing is truly separated from the development team, and it usually has a separate head count and a separate budget. However, the real user acceptance testing is executed by the client's special user group. The user group consists of people from different backgrounds, such as software quality assurance engineers, business associates, and customer support engineers including the major stakeholders of the customer organization.

Conclusion and Future Scope

The paper covers the concept of testing, its role in software development cycle, methods of testing, level of testing and preparing test cases. In the current competitive era, for a software product to sustain in market for long time, it is mandatory to have good quality and value. Following the test driven environment can help to work testing first mean while the production code is being written.

ACKNOWLEDGMENT

I would like to thank Charotar University of Science and Technology to provide with all the resources required to carry out the work in this paper.

REFERENCES

[1] P. Ron. Software testing. Vol. 2. Indianapolis: Sam's, 2001.

[2] S. Amland, "Risk-based testing:" Journal of Systems and Software, vol. 53, no. 3, pp. 287–295, Sep. 2000.

[3] Redmill and Felix, "Theory and Practice of Risk-based Testing", Software Testing, Verification and Reliability, Vol. 15, No. 1, March 2005.

[4] B. Agarwal et al., "Software engineering and testing". Jones & Bartlett Learning, 2010.

[5] Mailewa, Akalanka, Jayantha Herath, and Susantha Herath. "A Survey of Effective and Efficient Software Testing." The Midwest Instruction and Computing Symposium. Retrieved from http://www.micsymposium. org/mics2015/ProceedingsMICS_2015/Mailewa_ 2D1_41. pdf. 2015.

[6] Mohd. Ehmer Khan, "Different Approaches to White Box testing Technique for Finding Errors," IJSEIA, Vol. 5, No. 3, pp 1-13, July 2011.

[7] Mohd. Ehmer Khan, "Different Approaches to Black Box Testing Technique for Finding Errors," IJSEA, Vol. 2, No. 4, pp 31-40, October 2011.

[8] L. Osterweil. Strategic directions in software quality. ACM Computing Surveys, 4:738–750, Dec. 1996.

[9] A. Bertolino. Software testing research: Achievements, challenges, dreams. In 2007 Future of Software Engineering, pages 85–103, 2007.

[10] M. J. Harrold. Testing: A roadmap. In Proceedings of the Conference on the Future of Software Engineering, pages 61–72, 2000.

[11] M.Kumar, S.K.Singh,. R.K.Drivedi, "A Comparative Study of Black Box Testing and White Box Testing Techniques", International Journal of Advance Research in Computer Science and Management Studies, Volume-3, Issue 10,pp. 32-44,October 2015, ISSN: 2321-7782

[12] M.E. Khan, "Different Approaches to Black Box Testing Technique for Finding Errors", IJSEA, Volume- 2, Issue- 4, pp 31-40, October 2011.

[13] Oluigbo I.V., Asiegbu B.C., Ezeh G.N., Nwokonkwo O.C., "Group Membership Prediction of an Outsourced Software Project: A Discriminant Function Analysis Approach", International Journal of Scientific Research in Multidisciplinary Studies , Vol.3, Issue.4, pp.12-18, 2017.

[14] Suresh Jat and Pradeep Sharma, "Analysis of Different Software Testing Techniques ",International Journal of Scientific Research in Computer Science and Engineering Vol.5, Issue.2, pp.77-80, April 2017.

[15] Chandraprakash Patidar ,"Test Case Generation Using Discrete Particle Swarm Optimization Algorithm", International Journal of Scientific Research in Computer Science and Engineering ISSN 2320-7639 Volume-1, Issue-1, Jan- Feb-2013.

Authors Profile

Nirali Honest is working as an Assistant Professor, at Smt. Chandaben Mohanbhai Patel Institute of Computer Applications, since August 2005. She has completed her Master of Computer Applications from Gujarat University in 2005 and Bachelor of Computer Applications in 2002 form Dharamsinh Desai University. During her tenure at academics she has conducted sessions on various courses like Software Quality Assurance, Software Engineering and Quality Assurance, Web Technologies and Applications, Database Management Concepts, System Analysis and designing , Network Technologies, Object Oriented Designing and Modeling, Parallel Processing, Operating Systems, Object Oriented concepts and programming in java, Computer based management systems, Data Structures, Fundamentals of Commerce and Business Processes. She has completed her Ph.D. in Computer Science and Applications in November,2016, her area of work is Web Usage Mining. She has published/presented thirteen research papers in national/ international journals/conferences of repute.

