Volume 5, Issue 11, November 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Software Testing and Evaluation

Krishna Kumar Yadav
Student (M.tech), Department of Computer Science and Engineering
Kanpur Institute of Technology, Rooma, Kanpur,
Uttar Pradesh, India

Abstract - Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not. Testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements. This tutorial will give you a basic understanding on software testing, its types, methods, levels, and other related terminologies.

Keywords - Software quality assurance, STLC, SWEBOK, Extreme Programming (XP), integration and testing (I&T), ICT.

I. INTRODUCTION

Software testing is an activity that helps in finding out bugs, defects or errors in a software system under development. Software testing is an activity in which a system or component is executed under specified conditions; the results are observed and recorded and an evaluation is made of some aspects of the system or component. Software testing is a critical element of software quality assurance. It represents the ultimate process to ensure the correctness of the product. It enhances customer confidence and there by increases the business economics. Testing the product means adding value to it that means raising the quality or reliability of the program. Raising the reliability of the product means finding and removing errors. Testing is the process of executing a program with the intent of finding errors.

II. IMPORTANCE OF SOFTWARE TESTING

Software Testing is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expensive or dangerous. We need to check everything and anything we produce because things can always go wrong - humans make mistakes all the time. Since we assume that our work may have mistakes, hence we all need to check our own work. However some mistakes come from bad assumptions and blind spots, so we might make the same mistakes when we check our own work as we made when we did it. So we may not notice the flaws in what we have done. Ideally, we should get someone else to check our work because another person is more likely to spot the flaws. There are several reasons which clearly tell us as why Software Testing is important and what are the major things that we should consider while testing of any product or application.

Software testing is very important because of the following reasons:

- Software testing is really required to point out the defects and errors that were made during the development phases.
- It's essential since it makes sure of the Customer's reliability and their satisfaction in the application.
- It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence.
- Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.
- Testing is required for an effective performance of software application or product.
- It's important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development.

III. SOFTWARE TESTING LIFE CYCLE

Software Testing Life Cycle (STLC) refers to a testing process which has specific steps to be executed in a definite sequence to ensure that the quality goals have been met. In STLC process, each activity is carried out in a planned and systematic way. Each phase has different goals and deliverables. Different organizations have different phases in Software Testing Life Cycle; however the basis remains the same.

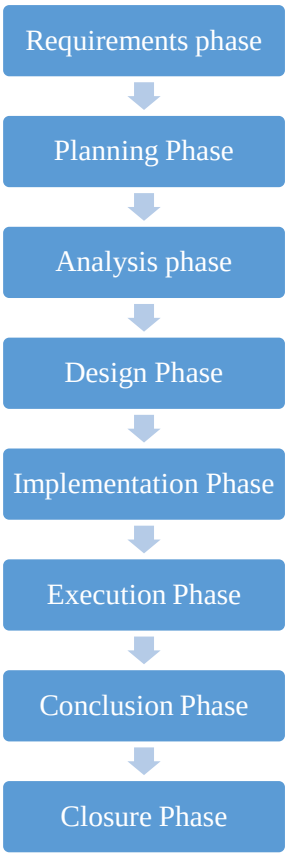

Fig. 1 Different phases involved in software test life cycle

A. Requirement Phase: During this phase of Software Testing Life Cycle, analyze and study the requirements. Have brain storming sessions with other teams and try to find out whether the requirements are testable or not. This phase helps to identify the scope of the testing. If any feature is not testable, communicate it during this phase so that the mitigation strategy can be planned.

B. Planning Phase: In practical scenarios, Test planning is the first step of the testing process. In this phase we identify the activities and resources which would help to meet the testing objectives. During planning we also try to identify the metrics, the method of gathering and tracking those metrics.

C. Analysis Phase: This STLC phase defines "WHAT" to be tested. We basically identify the test conditions through the requirements document, product risks and other test basis. The test condition should be traceable back to the requirement. There are various factors which effect the identification of test conditions

D. Design Phase: This phase defines "HOW" to test. This phase involves the following tasks:

- Detail the test condition. Break down the test conditions into multiple sub conditions to increase coverage.
- Identify and get the test data.
- Identify and set up the test environment.
- Create the requirement traceability metrics.
- Create the test coverage metrics.

E. Implementation Phase: The major task in this STLC phase is of creation of the detailed test cases. Prioritize the test cases also identify which test case will become part of the regression suite. Before finalizing the test case, It is important to carry out the review to ensure the correctness of the test cases. Also don't forget to take the sign off of the test cases before actual execution starts. If your project involves automation, identify the candidate test cases for automation and proceed for scripting the test cases. Don't forget to review them!

F. Execution Phase: As the name suggests, this is the Software Testing Life Cycle phase where the actual execution takes place. But before you start your execution, make sure that your entry criterion is met. Execute the test cases, log defects in case of any discrepancy. Simultaneously fill your traceability metrics to track your progress.

*G. Conclusion Phase***:** This STLC phase concentrates on the exit criteria and reporting. Depending on your project and stakeholders choice, you can decide on reporting whether you want to send out a daily report of weekly report etc. There are different types of reports (DSR – Daily status report, WSR – Weekly status reports) which you can send, but the important point is, the content of the report changes and depends upon whom you are sending your reports. If Project managers belong to testing background then they are more interested in the technical aspect of the project, so include the technical things in your report (number of test cases passed, failed, defects raised, severity 1 defects etc.). But if you are reporting to upper stakeholders, they might not be interested in the technical things so report them about the risks that have been mitigated through the testing.

H. Closure Phase: Tasks for the closure activities include the following:

- Check for the completion of the test. Whether all the test cases are executed or mitigated deliberately. Check there are no severity 1 defects opened.
- Do lessons learnt meeting and create lessons learnt document.(Include what went well, where are the scope of improvements and what can be improved)

IV. LEVELS OF SOFTWARE TESTING

There are generally four recognized levels of software testing's: unit testing, integration testing, system testing and acceptance testing. Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. The main levels during the development process as defined by the SWEBOK guide are unit, integration, and system testing that are distinguished by the test target without implying a specific process model. Other test levels are classified by the testing objective.

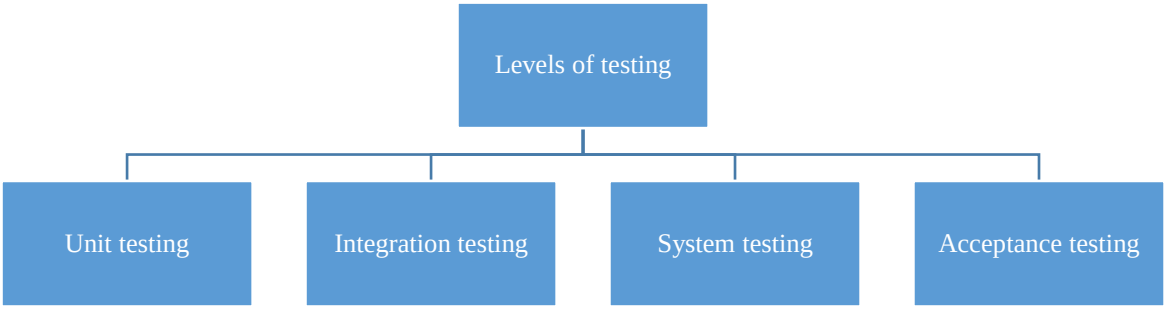

Fig. 2 Various levels of software testing

A. Unit testing: Unit testing is a software development process in which the smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation. Unit testing is often automated but it can also be done manually. The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

B. Integration testing: Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing. Integration testing can expose problems with the interfaces among program components before trouble occurs in real-world program execution. Integration testing is a component of Extreme Programming (XP), a pragmatic method of software development that takes a meticulous approach to building a product by means of continual testing and revision.

C. System Testing: System Testing is a level of the software testing where complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. It may include tests based on risks and/or requirement specifications, business process, use cases, or other high level descriptions of system behavior, interactions with the operating systems, and system resources. System testing is most often the final test to verify that the system to be delivered meets the specification and its purpose.

D. Acceptance testing: It is nothing but a system level testing with the only difference that it is carried out in the presence of the user. It falls under the category of black box testing. Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

V. METHODS OF SOFTWARE TESTING

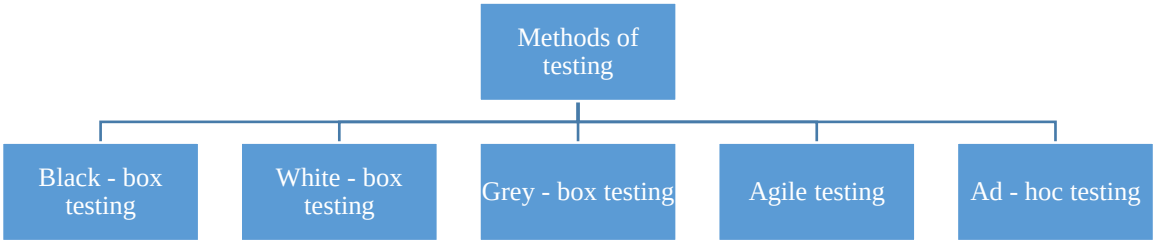

Fig. 3 Different methods of software testing

A. Black-box testing: Black-box testing, also called functional or specification-based testing. In black-box testing, test cases are derived from the specification of the software, i.e. we do not consider implementation details. Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well.

B. White-box testing: White-box testing, also called structural or program-based testing. This is a complementary approach, in which we do consider the internal logical structure of the software in the derivation of test cases. White-box testing is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). White-box testing can be applied at the unit, integration and system levels of the software testing process. Although traditional testers tended to think of white-box testing as being done at the unit level, it is used for integration and system testing more frequently today. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it has the potential to miss unimplemented parts of the specification or missing requirements.

C. Grey Box testing: Grey Box testing is testing technique performed with limited information about the internal functionality of the system. Grey Box testers have access to the detailed design documents along with information about requirements. Grey Box tests are generated based on the state-based models, UML Diagrams or architecture diagrams of the target system.

D. Agile testing: Agile testing is a software testing practice that follows the principles of agile software development. Agile testing involves all members of a cross-functional agile team, with special expertise contributed by testers, to ensure delivering the business value desired by the customer at frequent intervals, working at a sustainable pace. Specification is used to capture examples of desired and undesired behavior and guide coding.

E. Ad hoc testing: Ad hoc testing is a commonly used term for software testing performed without planning and documentation, but can be applied to early scientific experimental studies. The tests are intended to be run only once, unless a defect is discovered. Ad hoc testing is the least formal test method. As such, it has been criticized because it is not structured and hence defects found using this method may be harder to reproduce (since there are no written test cases). However, the strength of ad hoc testing is that important defects can be found quickly. It is performed by improvisation: the tester seeks to find bugs by any means that seem appropriate. Ad hoc testing can be seen as a light version of error guessing, which itself is a light version of exploratory testing.

VI. FUNCTIONAL AND NON FUNCTIONAL TESTING

A. Functional Testing: Functional Testing is the type of testing done against the business requirements of application. It is a black box type of testing. It involves the complete integration system to evaluate the system's compliance with its specified requirements. Based on the functional specification document this type of testing is to be carried out. In actual testing, testers need to verify a specific action or function of the code. For functional testing either manual testing or automation tools can be used but functionality testing would be easier using manual testing only. Prior to non Functional testing the Functional testing would be executed first.

Five steps need to be keeping in mind in the Functional testing:

- Preparation of test data based on the specifications of functions.
- Business requirements are the inputs to functional testing.
- Based on functional specifications find out of output of the functions.
- The execution of test cases.
- Observe the actual and expected outputs.

B. *Non Functional Testing:* The non Functional Testing is the type of testing done against the non functional requirements. Most of the criteria are not consider in functional testing so it is used to check the readiness of a system. Non-functional requirements tend to be those that reflect the quality of the product, particularly in the context of the suitability perspective of its users. It can be started after the completion of Functional Testing. The non functional tests can be effective by using testing tools. The testing of software attributes which are not related to any specific function or user action like performance, scalability, security or behavior of application under certain constraints. on functional testing has a great influence on customer and user satisfaction with the product. Non functional testing should be expressed in a testable way, not like "the system should be fast" or "the system should be easy to operate" which is not testable. Basically in the non functional test is used to major non-functional attributes of software systems. Let's take non functional requirements examples; in how much time does the software will take to complete a task? Or how fast the response is.

VII. CONCLUSION

In this paper, a comprehensive analysis on software testing is provided. The levels and classification of software testing has been discussed. The review of different approaches of software testing has been presented.

REFERENCES

[1] Laycock, G. T. (1993*). "The Theory and Practice of Specification Based Software Testing"* (PostScript). Dept of Computer Science, Sheffield University, UK. Retrieved 2008-02-13.

[2] Savenkov, Roman (2008). *How to Become a Software Tester.* Roman Savenkov Consulting. p. 159. ISBN 978-0-615-23372-7.

[3] Kolawa, Adam; Huizinga, Dorota (2007). *Automated Defect Prevention: Best Practices in Software Management.* Wiley-IEEE Computer Society Press. pp. 41–43. ISBN 0-470-04212-5.

[4] Section 1.1.2, Certified Tester Foundation Level Syllabus, International Software Testing Qualifications Board.
[5] "*Proceedings from the 5th International Conference on Software Testing and Validation (ICST).*" Software Competence Center Hagenberg. *"Test Design: Lessons Learned and Practical Implications."*
[6] *"SOA Testing Tools for Black, White and Gray Box SOA Testing Techniques"*. Crosschecknet.com. Retrieved 2012-12-10.
[7] Tran, Eushiuan (1999). *"Verification/Validation/Certification"*. In Koopman, P. Topics in Dependable Embedded Systems. USA: Carnegie Mellon University. Retrieved 2008-01-13.
[8] Paul Krill (22 August 2014). *"Software testers balk at ISO 29119 standards proposal"*. InfoWorld.
[9] *"IEEE article on Exploratory vs. Non Exploratory testing"* (PDF). Ieeexplore.ieee.org. Retrieved 2012-01-13.
[10] IEEE (1990). IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries. New York: IEEE. ISBN 1-55937-079-3.
[11] Part of the Pipeline: Why Continuous Testing Is Essential, by Adam Auerbach, TechWell Insights August 2015.
[12] The Relationship between Risk and Continuous Testing: An Interview with Wayne Ariola, by Cameron Philipp-Edmonds, Stickyminds December 2015.
[13] ISO/IEC/IEEE 29119-1:2013 – Software and Systems Engineering – Software Testing – Part 1 – Concepts and Definitions; Section 4.38.
[14] Pan, Jiantao (Spring 1999). *"Software Testing (18-849b Dependable Embedded Systems)"*. Topics in Dependable Embedded Systems. Electrical and Computer Engineering Department, Carnegie Mellon University.
[15] Rodríguez, Ismael (2009). *"A General Testability Theory"*. CONCUR 2009 - Concurrency Theory, 20th International Conference, CONCUR 2009, Bologna, Italy, September 1–4, 2009.
[16] Microsoft Development Network Discussion on exactly this topicArchived April 2, 2015 at the Wayback Machine.
[17] Black, Rex (December 2008). *Advanced Software Testing- Vol. 2:* Guide to the ISTQB Advanced Certification as an Advanced Test Manager. Santa Barbara: Rocky Nook Publisher. ISBN 1-933952-36-9.
[18] EtestingHub-Online Free Software Testing Tutorial. *"e)Testing Phase in Software Testing:"*. Etestinghub.com. Retrieved 2012-01-13.
[19] Myers, Glenford J. (1979). *The Art of Software Testing*. John Wiley and Sons. pp. 145–146. ISBN 0-471-04328-1.
[20] *"Globalization Step-by-Step: The World-Ready Approach to Testing. Microsoft Developer Network"*. Msdn.microsoft.com. Retrieved 2012-01-13.

