Digital Innovations & Contemporary Research in SCIENCE, ENGINEERING & TECHNOLOGY

Vol. 5 No. 4. December, 2017

Article Citation Format
S.O. Akinola & F.O. Ibitowa (2017): A Review on Software Testing.
Journal of Digital Innovations & Contemp Res. In Sc., Eng & Tech.
Vol. 5, No. 4. Pp 111-120

Article Progress Time Stamps

Article Type: Research Article
Manuscript Received: 19th August, 2017
Review Type: Blind - Expedited
Final Acceptance:: 21st December, 2017
DOI Prefix: 10.22624

A Review on Software Testing

[1]S.O. Akinola & [2]F.O. Ibitowa
[1&2]Department of Computer Science,
[1]University of Ibadan, Ibadan, Nigeria
[2]The Polytechnic, Ibadan, Ibadan, Nigeria.
E-mail: solom202@yahoo.co.uk, ibitowafolashade@yahoo.com
Phone: 08056666117, 08062334553

ABSTRACT

SDLC, System Development Life Cycle is a process used by software industry to design, develop, test and deploy high quality software. The SDLC aims to produce a high quality software that meets or exceeds customer requirements and expectations, reaches completion within times and cost estimates. System Development Life Cycle stages consists of Requirement stage, Design stage, Build (Code Writing) stage, Test stage and Deploy and Maintenance stage. According to this SDLC stages, Software testing is one of the stages and it means checking if a program for specified inputs gives correctly and expected results. We present the current knowledge of software testing based on existing literature. Software testing is an activity which is aimed for evaluating quality of a program and also for improving it, by identifying defects and problems. It is a process of verifying and validating that a software application or program meets the business and technical requirements that guided its design and development and works as expected. Hence, the goal of testing is systematical detection of different classes of errors in a minimum amount of time and with a minimum amount of effort. The main purpose of testing are Quality Assurance, Reliability estimation, defects finding, validation and verification. Software Testing is also used to test the software for other software quality factors like Reliability, Usability, Integrity, Security, Capability, Efficiency, Portability, maintainability, Compatibility. This paper looks at the significance and the basic concepts of Software testing as a process needful in deploying quality and better software at the right time with minimum effort.

Keywords: SDLC, Software Testing, Verifying, Validating, Quality Assurance, requirements1.

1. INTRODUCTION

The overall aim of the software industry is to ensure delivery of high quality software to the end user. To ensure high quality software, it is required to test software. Testing ensures that software meets user specifications and requirements (Chayanika al, 2013).

Systems Development Life Cycle (SDLC), or Software Development Life Cycle refers to the process of developing a new or amending existing Software, and the methodologies employed to develop these systems. It is a formalized process of building information systems in a very deliberate, structured and methodical way, reiterating each stage of the life cycle. System Development Life Cycle stages consists of Requirement stage, Design stage, Build (code writing) stage, Test stage and Deploy and Maintenance stage.

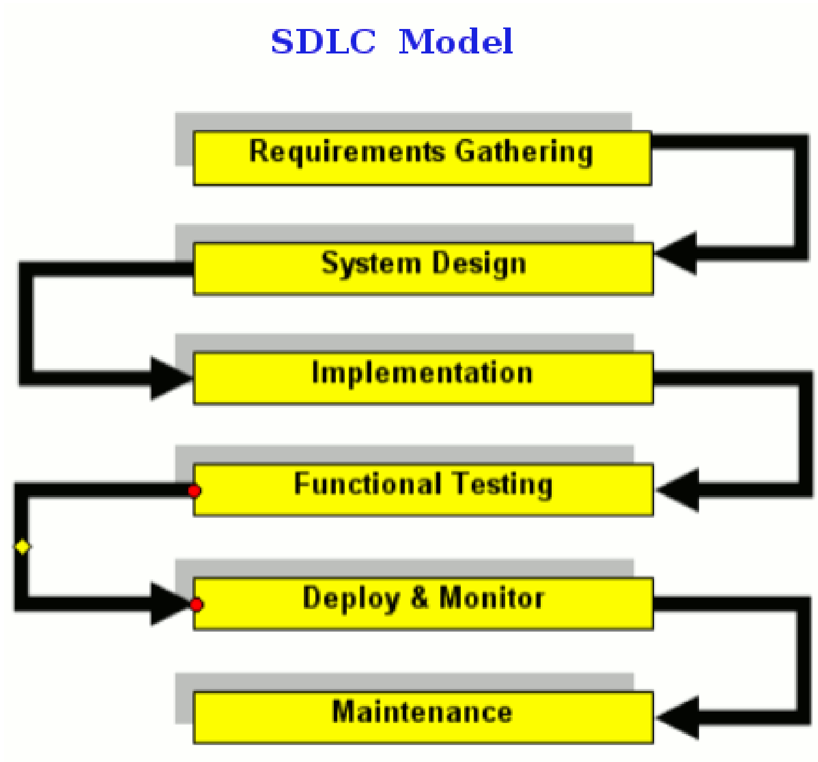

Fig. 1: The Software Development Lifecycle Model

Software testing is one of the stages of the SDLC. Testing means checking if a program for specified inputs gives correctly and expected results. Hence, the goal of testing is systematical detection of different classes of errors in a minimum amount of time and with a minimum amount of effort. The main purpose of testing are Quality Assurance, Reliability estimation, defects finding, validation and verification. Testing is also used to test the software for other software quality factors like Reliability, Usability, Integrity, Security, Capability, Efficiency, Portability, Maintainability, and Compatibility.

Developers can build powerful, sophisticated applications, and they can also build applications that frustrate users, waste computer resources and damage the credibility of the developer and software. Therefore, it has become imperative to test the software components and software as a whole for integrity, capability, reliability, efficiency, portability, maintainability, compatibility and usability of the software. The only way to ensure that application software are secure and stable is to rely on software testing. Testing is done manually or using automated tools. Testing is done right from the beginning of the Software Development Life Cycle till the end, it is delivered to the customer.

The rest of the paper is structured as follows. In the next section we describe current researches in software testing, followed by the general concepts of software testing. The paper is rounded up with conclusion and future work.

Definition of Terms

Errors: These are actual coding mistakes made by developers. In addition, the difference in output of software and desired output, is considered an error.

Fault: When error exists fault occurs. A fault is a result of an error which can cause system to fail. A fault is the result of an error in the software documentation code etc.

Failure: - failure is said to be the inability of the system to perform the desired task. Failure occurs when fault exists in the system.

2. Current Researches in Software Testing

The context of this study is to present the current knowledge of software testing based on existing literature. Thus, in this section we review the related works in the software testing literature. Abhijit (2012) worked on "Software Testing techniques and Strategies". The paper presents Software testing, need for software testing, Software testing goals and principles. It also describe different Software testing techniques and different software testing strategies. Lastly it describes the difference between software testing and debugging.

In the work of Jovanovic (2008) on "Software Testing Methods and Techniques", main testing methods and techniques are shortly described. General classification is outlined: two testing methods – black box testing and white box testing, and their frequently used techniques. Black box techniques: Equivalent partitioning, Boundary Value Analysis, Cause Effect Graphing Techniques and Comparison Testing. White box techniques: Basis Path testing, Loop testing and Control Structure Testing. Also the classification of the IEEE Computer Society is illustrated.

Mohd. (2011) worked on "Different Approaches to White Box Testing Techniques for finding Errors". According to this paper, Software testing is the process of analyzing software to find the difference between required and existing condition. Software testing is performed throughout the development cycle of software and it is also performed to build quality software, for this purpose two basic testing approaches are used, they are white box testing and black box testing. One of the software testing techniques explained is Black box testing. It is a method of generating test cases that are independent of software internal structure. The paper also explores various different approaches to black box testing technique for finding errors. Since black box testing is always based either directly or indirectly on the software specification so it is also called specification based testing.

Sheetal, Savita and Chawan (2012) on "Software Testing Strategies and Techniques", discusses different software strategies such as unit testing, integration testing, validation testing and techniques such as white box and black box testing for conventional and object oriented software development. Strategy provides a guideline to conduct testing. The testing carried out at different stages in software development life cycle is described by the various testing methods. Umar, Ashutosh and Priyaranjan (2013) on "Software Testing", explain that Software testing is the process of validating and verifying that a software program/applications meets that requirement that guide its design and development, work as expected and can be implemented with same characteristics. Many attempts to obtain software testing based on methods from the field of design of experiments have been recommended as a means of providing high coverage at relatively low cost. Many tools to generate aim pairs or higher e-degree combination, of input values have been developed and demonstrated in some applications, but little empirical evidence is available to help developer in evaluating the effectiveness of these tools for particular complexities.

In the work of Rajkumar and Alagarsamy (2013), on "The most common factors for the failure of software development project", assert that software projects fail completely or partial failures because a small number of projects meet all their requirements. These requirements can be the cost, schedule, quality, or requirements objectives. The paper further explains that a major part of the failure can be due to a lack of understanding of the software development process and the effect of that method used in the project plan, schedule and cost estimates. The paper also presents that organizations and individuals have studied a number of projects that have both succeeded and failed and some common factors emerge. A key finding in the paper is that there is no one overriding factor that causes project failure. This paper review the numbers of factors that are involved in project failure, some of which interact with each other.

SMK Quadri and Sheik (2010), on "Software Testing – Goals, Principles and limitations", describes Software testing as an activity which is aimed for evaluating quality of a program and also for improving it, by identifying defects and problems. Software testing strives for achieving its goals (both implicit and explicit) but it does have certain limitations. The paper further explains that testing can be done more effectively if certain established principles are followed. In spite of having limitations software testing continues to dominate other verification techniques like static analysis, model checking and proofs. So it is indispensable to understand the goals, principles and limitations of software testing so that the effectiveness of software testing could be maximized.

2.1 Software Testing

There are several definitions of software testing:
Software Testing is the process of validating and verifying that a software meets that requirements that guide its design and development, work as expected and can be implemented with same characteristics (Uma Nath Yadav et al, 2013).

Software Testing is a process of finding errors or bug in the software and it involves any activity and evaluating an attribute or capabilities of a program or software and determining that it meets its required results (Uma Nath Yadav et
al, 2013).

Software Testing is a process of executing a program with the goal of finding errors. (Sheetal Thakare et al, 2012). Software Testing is an activity performed for evaluating software quality and for improving it. (Jovanovic Irena, 2008). Software Testing is a process of verifying and validating that a software application or program meets the business and technical requirements that guided its design and development and works as expected. (John E.B. et al,). Software Testing is the process of analyzing a software item to detect the differences between existing and required conditions and to evaluate the features of the software item (Laurie Williams, 2006).

2.1.1 Qualities of Software Products

a. Completeness refers to the availability of all features listed in the requirements, or in the user manual. An incomplete is one that does not fully implement all features required.
b. Consistency refers to adherence to a common set of conventions and assumptions. For example, all buttons in the user interface might follow a common color coding convention
c. Reliability is the probability of failure free operation of software in its intended environment
d. Robustness is the ability of a software product to cope with unusual situation.
e. Usability refers to the ease with which an application can be used. The amount of efforts or time required to learn how to use the software should be less. This makes the software user-friendly even for IT-illiterate people.
f. Performance refers to the time the application takes to perform a requested task

g. Portability is the ability of software to perform same functions across all environments and platforms, demonstrate its portability.
h. Reusability : If we are able to use the software code with some modifications for different purpose then we call software to be reusable
i. Interoperability is the ability of software to exchange information with other applications and make use of information transparently
j. Maintainability means maintenance of the software should be easy for any kind of user.
k. Modularity : Any software is said to made of units and modules which are independent of each other. These modules are then integrated to make the final software. If the software is divided into separate independent parts that can be modified, tested separately, it has high modularity.

2.1.2 Problems of Software Development

1. Lateness in software delivery, over budget, and of unsatisfactory quality;
2. Software validation and verification are rarely systematic and are usually not based on sound and well-defined techniques;
3. Software development processes are commonly unstable and uncontrolled;
4. Software quality is poorly measured, monitored and controlled. (Andrea Arcuri, 2009)

2.1.3 Causes of Software Failure

1. The specification may be wrong or have a missing requirement;
2. The specification may contain a requirement that is impossible to implement given the prescribed software and hardware;
3. The system design may contain a fault;
4. The program code may be wrong;
5. Lack of compatibility with current development in hardware equipment.

2.1.4 Effects of Software Failure

1. Space Applications: Loss of lives and launch delays. Due to a malfunction in the control software, the rocket veered off its flight path 37 seconds after launch.;
2. Defense and Warfare: Misidentification of friend or foe;
3. Transportation: Deaths, delays, sudden acceleration and inability to brake;
4. Electric Power: Death, injuries, power outages and long-term health hazards (radiation);
5. Money Management: Fraud, violation of privacy, shutdown of stock exchanges and banks negative interest rates;
6. Control of Elections: Wrong results (intentional or non-intentional);
7. Control of Jails: Technology-aided escape attempts and successes, failures in software-controlled locks;
8. Law Enforcement: False arrests and imprisonments (Andrea Arcuri,2009).

The following professionals are involved in testing of a system within their respective capacities:

i. Software Tester
ii. Software Developer
iii. Project Lead/Manager
iv. End User
v. Ethical Hackers

Different companies have different designations for people who test the software on the basis of their experience and knowledge such as Software Tester, Software Quality Assurance Engineer, and Quality Assurance Analyst etc.

2.2 Types of Software Testing

There are two types of software testing

1. Manual Testing
2. Automated Testing

Manual Testing- This testing is performed without taking help of automated testing tools. The software tester prepares test cases for different sections and levels of the code, executes the tests and reports the result to the manager. Manual testing is time and resource consuming. The tester needs to confirm whether or not right test cases are used. Major portion of testing involves manual testing.
Automated Testing - This testing is a testing procedure done with the aid of automated testing tools. The limitations with manual testing can be overcome using automated test tools.

2.3 Software Testing Methods

There are different methods which can be used for Software testing.

2.31 Black Box Testing

The technique of testing without having any knowledge of the interior workings of the application is Black Box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.
Black box Testing is synonymous to

i. Opaque Testing
ii. Functional Testing
iii. Behavioural Testing
iv. Closed box testing

Advantages:

1. Well suited and efficient for large code segments.
2. Code Access not required.
3. Clearly separates user's perspective from the developer's perspective through visibly defined roles.
4. Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems.

Disadvantages:

1. Limited Coverage since only a selected number of test scenarios are actually performed.
2. Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
3. Blind Coverage, since the tester cannot target specific code segments or error prone areas.
4. The test cases are difficult to design.

2.3.2 White Box Testing

White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also called glass testing or open box testing. In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.
White box Testing is synonymous to

i. Glass box testing
ii. Clear box Testing
iii. Open box Testing

iv. Transparent box Testing
v. Structural Testing
vi. Logic driven testing
vii. Design based Testing
viii. Specification Based Testing

Advantages:
1. It becomes very easy to find out which type of data can help in testing the application effectively since the tester has knowledge of the source code;
2. It helps in optimizing the code;
3. Extra lines of code can be removed which can bring in hidden defects;
4. Maximum coverage is attained during test scenario writing due to the tester's knowledge about the code.

Disadvantages:
1. The costs are increased due to the fact that a skilled tester is needed to perform white box testing,.
2. Sometimes it is impossible to look into every nook and cranny of the software to find out hidden errors that may create problems as many paths will go untested;
3. It is difficult to maintain white box testing as the use of specialized tools like code analyzers and debugging tools are required.

2.3.3 Grey Box Testing

Grey Box testing is a technique to test the application with limited knowledge of the internal workings of an application. In software testing, the term "the more you know the better" carries a lot of weight when testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black box testing, where the tester only tests the application's user interface, in grey box testing, the tester has access to design documents and the database. Having this knowledge, the tester is able to prepare better test data and test scenarios when making the test plan.

Advantages:
1. Offers combined benefits of black box and white box testing wherever possible;
2. Grey box testers don't rely on the source code, instead they rely on interface definition and functional specifications.
3. Based on the limited information available, a grey box tester can design excellent test scenarios especially around communication protocols and data type handling.
4. The test is done from the point of view of the user and not the designer.

Disadvantages:
1. The ability to go over the code and test coverage is limited since the access to source code is not available,.
2. The tests can be redundant if the software designer has already run a test case.
3. Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many program paths will go untested.

2.4 Software Testing Techniques

Black Box

i. Equivalent Partitioning
ii. Boundary value Analysis
iii. Cause-Effect Graphing
iv. Comparison Testing
v. Fuzz Testing
vi. Model-Based Testing

White Box

i. Basis Path Testing
ii. Control Flow testing
iii. Data Flow testing
iv. Branch Testing
v. Loop testing
vi. Control Structure Testing

2.5 Levels of Testing

1. Unit Testing - is a level of the software testing process where individual units/ components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.
2. Integration Testing- : The testing of individual units helps in removing local faults, but does not exercise the interactions among different units. Integration testing is the activity of exercising such interactions by pulling together the different modules composing a system. It is characterized by involving different interacting units which have been in general developed by different programmers. In this case the code is still visible, but with a higher granularity.
3. System Testing - is the testing of the system as a whole. It is characterized by being performed on a code which is in general not visible, due to both accessibility and complexity reasons. This kind of test addresses all the properties of software that cannot be expressed in terms of the properties of the subsystems composing it. At this level the software behavior is compared with the expected one according to the specifications.
4. Acceptance Testing – is the set of activities performed by the end-user whose goal is to verify whether the system is behaving as it was intended to. This level is characterized by the absence of visibility of specification and design documents.
5. Regression Testing - Regression testing is performed during maintenance. As a system is used, it often requires to be modified to correct faults, to enrich its functionalities, or to adapt it to environmental changes. Modifications performed on system imply the need for re-testing it, even when small changes are concerned. Such activity is known as regression testing.

2.6 The Economics of Software Testing

In software development, there are costs associated with testing programs. There is need to write out test plan and test cases, to set up the proper equipment, to systematically execute the test cases, to follow up on problems that are identified, and to remove most of the faults we find. Actually, sometimes we can find low-priority faults in our code and decide that it is too expensive to fix the fault because of the need to redesign, recode, or otherwise remove the fault. These faults can remain latent in the product through a follow-on release or perhaps forever (Laurie Williams, 2006).

For faults that are not discovered and removed before the software has been shipped, there are costs. Some of these costs are monetary, and some could be significant in less tangible ways. Customers can lose faith in the business and can get very angry. They can also lose a great deal of money if their system goes down because of the defects. (Think of the effect on a grocery store that can't check out the shoppers because of its "down" point-of-sale system.) And, software development organizations have to spend a great deal of money to obtain specific information about customer problems and to find and fix the cause of their failures. Sometimes, programmers have to travel to customer locations to work directly on the problem. These trips are costly to the development organization, and the customers might not be overly cheerful to work with when the programmer arrives. When we think about how expensive it is to test, we must also consider how expensive it is not to test – including these intangible costs as well as the more obvious direct costs (Laurie Williams, 2006).

We also need to consider the relative risk associated with a failure depending upon the type of project we work on. Quality is much more important for safety- critical software, like aviation software, than it is for video games. Therefore, when we balance the cost of testing versus the cost of software failures, we will test aviation software more than we will test video games. As a matter of fact, safety-critical software can spend as much as three to five times as much on testing as all other software engineering steps combined (Laurie Williams, 2006).

To minimize the costs associated with testing and with software failures, a goal of testing must be to uncover as many defects as possible with as little testing as possible. In other words, we want to write test cases that have a high likelihood of uncovering the faults that are the most likely to be observed as a failure in normal use. It is simply impossible to test every possible input-output combination of the system; there are simply too many permutations and combinations. As testers, we need to consider the economics of testing and strive to write test cases that will uncover as many faults in as few test cases as possible (Laurie Williams, 2006).

3. CONCLUSION AND FUTURE WORK

The purpose of this paper was to study what the current knowledge of software testing is in literature, what and why of software testing. We presented the current knowledge of Software testing, Software problems, its causes, its effects, basic concept of software testing, and economic impact of software testing based on the existing literature. We also looked at the perceived benefits and shortcomings of software testing.

Our study did not reveal any testing method or techniques that is best for testing software. We feel that further research on the best method or techniques for software testing is needed, because they might increase the effectiveness of software testing.

We plan to conduct more research to gain better understanding of software testing in software development.

REFERENCES

1. Abhijit A. Sawant, Pranit H. Bari and P.M. Chawan (2012), Software Testing Techniques and Strategies, IJERA International Journal of Engineering Research and Applications, Vol 2(3), pp.980-986
2. Andrea Arcuri (2009), Software Testing Overview, Simula Research Laboratory Oslo, Norway, http://www.uio.no/studier/emner/matnat/ifi/INF1050/v10/undervisningsmateriale/Testing_INF1050.pdf downloaded in May 2016.
3. Chayanika Sharmal, Sangeeta Sabharwal, Ritu Sibal (2013), A Survey on Software Testing Techniques using Genetic Algorithm, IJCSI International Journal of Computer Science Issues, Vol. 10, No. 1 pp.381-393.
4. Jovanovic Irena (2008), Software Testing Methods and Techniques, Manuscript, Div Inzenjering, d.o.o., Belgrade.
5. John E. Bentley (2004), Wachovia Bank, Charlotte NC, http://www2.sas.com/proceedings/sugi30/141-30.pdf downloaded in May 2016.
6. June Verner, Jennifer Sampson and Narciso Cerpa (2008), What factors lead to software project failure, Proceedings 2nd International Conference on Research Challenges in Information Science, IEEE, pp.71-80.
7. Laurie Williams (2006), Testing Overview and Black Box Testing Techniques, http://agile.csc.ncsu.edu/SEMaterials/BlackBox.pdf downloaded in May 2016.
8. Mohd. Ehmer Khan (2011). Different Approaches to White Box Testing Techniques for finding Errors, International Journal of Software Engineering and Its applications, Vol.5 No 3, July 2011.
9. Othmar Othmar Mwambe and Oleksandr Lutsaievskyi (2013), Selection and Application of Software Testing Techniques to Specific Conditions of Software Projects, International Journal of Computer Applications, Vol 70(18).
10. Praveen Ranjan Srivastava and Tai-hoon Kim (2009), Application of Genetic Algorithm in Software Testing, International Journal of Software Engineering and Its Applications Vol. 3 No 4, October 2009, pp. 87-95.
11. Rajkumar G. and Alagarsamy K. (2013), The most common factors for the failure of software development project, The International Journal of Computer Science and Applications (TIJCSA) Vol 1(11), pp 74-77.
12. Rupinder Kaur and Jyotsna Sengupta (2011), Software Process Models and Analysis on Failure of Software Development Projects, International Journal of Scientific & Engineering Research, Vol 2(2).
13. Sapna Varshney and Monica Mehrotra (2013), Search Based Software Test Data Generation for Structural Testing: A Perspective, ACM SIGSOFT Software Engineering Notes Vol 38 No 4, July 2013.
14. Sheetal Thakare, Savita Chavan, P.M. Chawan (2012), Software Testing Strategies and Techniques, International Journal of Emerging Technology and Advanced Engineering vol. 2(4).
15. SMK Quadri, Sheik Umar Farouq (2010), Software Testing – Goals, Principles and limitations, International Journal of Computer Applications Vol 6(9)
16. Umar Nath Yadav, Ashutosh Rai, Priyaranjan Verma (2013), Software Testing, International Journal of Computer Science and Information Technologies, Vol. 4(2) pp.306-308