A REVIEW ON SOFTWARE TESTING IN SOFTWARE ENGINEERING

Software testing importance, tools and techniques.

[1]Richa Patel,[2]Mansi Shah

[1]Master of Engineering,[2]Bachelor of Engineering
Information Technology,
Parul Polytechnic Institute, Baroda,India

Abstract: This study has been undertaken to investigate the determinants of Software testing in Software Engineering. A review are showed using Tools, Techniques and Theory model. Software Testing is a process of finding errors while executing a program so that we get a zero defect software.Software testing is an important means of accessing quality of software. Though a lot of advancements have been done in formal methods and verification techniques, still we need software to be fully tested before it could be handled to the customer side. Thus there are a number of testing techniques and tools made to accomplish the task. Software testing is an important area of research and a lot of development has been made in this field.The importance of testing is never more apparent than when some piece of software is released without having undergone sufficient vetting, the results of which can be bodily harm, large economic losses, and even diminished quality of life.Fortunately, increasingly innovative techniques exist for testing systems in different domains, which helps ensure software's reliability.

I. INTRODUCTION

1 Software Testing is an activity that is performed for evaluating software quality and also for improving it (Guide to the Software Engineering Body of Knowledge, Swebok – A project of the IEEE Computer Society Professional Practices Committee, 2004). Thus, the goal of testing is systematically and stepwise detection of different classes of errors within a minimum amount of time and also with a much less amount of effort. Software testing is also an important component of software quality assurance (SQA), and a number of software organizations are spending up to 40% of their resources on testing. There are four main objectives of testing (Myers, Glenford J.(1979), IBM Systems Research Institute, Lecturer in Computer Science, Polytechnic Institute of New York, The Art of Software Testing, by John Wiley & Sons, Inc.):

1.Detection: Various errors, defects, and deficiencies are detected. System capabilities and various limitations, quality of all components, the work products, and the overall system are calculated

2.Prevention: In this information to prevent or reduce the number of errors, to clarify system specifications and system performance is provided. Different ways to avoid risks and to tackle problems in the future are identified.

3.Demonstration: It shows how the system can be used with various acceptable risk. It also demonstrates functions with special conditions and shows how products are ready for integration or use.

4.Improving quality: By doing effective testing on software, errors can be minimized and thus quality of software is improved.

For life-critical software like flight control, testing can be much expensive as risk analysis is also involved. Risk analysis means the probability by which a software project can experience undesirable events, such as delays, schedule, outright cancellation and cost overruns and much more. So, a number of test cases and test plans are made in testing which means that the behavior of a program is inspected on a finite set of test cases i.e. inputs, execution preconditions, and also expected outcomes for a particular objective, such as to follow a particular program path or to verify compliance with a specific requirement, for which valued inputs are created. Practically, the set of test cases is considered to be infinite, thus theoretically there are a lot of test cases even for the smallest and simplest program (Stacey, D. A., Software Testing Techniques). In that case, testing could take a lot of time even months and months to execute. So, how to choose a proper set of test cases? Practically, various techniques are used, and some of them are also correlated with risk analysis, while others are correlated with test engineering expertise. The basic purpose of software testing is verification, validation and error detection in order to find various errors and problems – and the aim of finding those problems is to get them fixed. Software testing is more than just error detection. Software testing is done under controlled conditions for:

Verification: To verify if system behaves as specified. It is the checking and testing of items, which includes software, for conformance and consistency of software by evaluating the results against pre-defined requirements. In verification we ask a question, are we building the product right?

Validation: In this we check the system correctness which is the process of checking that what has beenspecified by user and what the user actually wanted. In validation we ask a question: Are we building the right system?

II. SOFTWARE TESTING STRATAGIES

A software testing strategy integrates various software test case design methods into a well planned series of steps that result in successful testing of software. Software testing strategies are thus important for testing. Software testing strategy is generally developed by testing specialist, project managers and software engineer. There are four software testing strategies:

Unit testing:

It is done at the lowest level. It tests the basic unit of software, which can be a module or component. Unit is the smallest module i.e. smallest set of lines of code which can be tested. Unit testing is just one of the levels of testing which contribute to make the big picture of testing a whole system. Unit testing is generally considered as a white box test class.

Integration Testing:

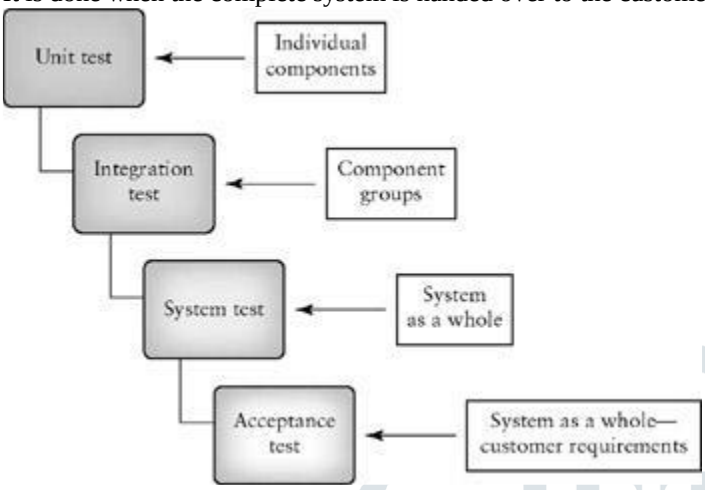
It is done when two or more tested units are combined into a larger structure. This testing is often done on the interfaces that are between the components and the larger structure that is being constructed, if its quality property cannot be properly assessed from its components.

System Testing:

It tends to test the end-to-end quality of the entire system. System test is often based on the functional and requirement specifications of the system. Non-functional quality attributes, such as security, reliability, and maintainability, are also checked.

Acceptance Testing:

It is done when the complete system is handed over to the customers or users from developer side.



The aim of acceptance testing is to give assure that the system is working rather than to find errors.

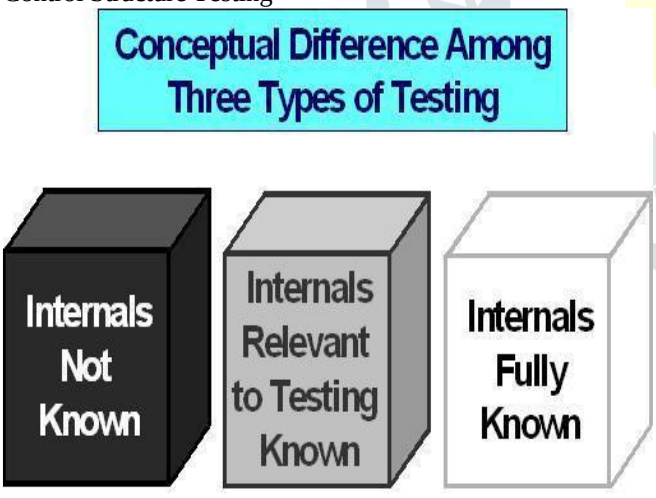
III. SOFTWARE TESTING METHODOLOGIES

There are following methodologies for software testing:

White Box Testing

In this testing, internal details and structure of system is made visible. Thus, it is highly efficient in detecting and resolving problems, because bugs can often be found before they cause trouble. We can thus define this method as testing software with the knowledge of its internal structure and coding. White box testing is also called clear box testing, white box analysis or clear box analysis. It is a strategy for finding errors in which the tester has complete knowledge of how the program components interact. This method is rarely used practically for debugging in large systems and networks, thus used for Web services applications. Different types of white box testing techniques are as follows:-

Basis Path Testing
Loop Testing
Control Structure Testing



Software testing methodologies

Black box testing

A black box is any device whose internal details and workings are not understood by or accessible to its user. It is testing of software based on specifications and output requirements and without any knowledge of the coding or internal structure in the program. The main aim is to test how well the system conforms to the specified requirements for the system. Black box testing have little or no knowledge to the internal logical structure of the system. Thus, it only examines the fundamental aspect of the system. It makes sure that all inputs are properly accepted and outputs are correctly produced (Software Engineering: A Practitioner's Approach, 6/e; Chapter 14: Software Testing Techniques, R.S. Pressman & Associates, Inc., 2005). Different types of Black box testing techniques are as follows:-

Equivalent Partitioning
Boundary value Analysis
Cause-Effect Graphing Techniques
Comparison Testing
Fuzz Testing

Model-based testing

Grey Box Testing

In recent years, a third testing method has been also considered i.e. grey box testing. It is defined as testing software and also having some knowledge of its internal logic and underlying code. It uses internal data structures and algorithms for designing the test cases more than black box testing but much less than white box testing. This method holds important when conducting integration testing between two or more modules of code written by different developers, where only their interfaces are exposed for testing (Redmill, Felix (2005), Theory and Practice of Risk-based Testing, Vol. 15, No. 1). This method includes reverse engineering to determine boundary values. Grey box testing is unbiased and non-intrusive because it doesn't require that the tester have access to internal source code.

IV. SOFTWARE TESTING TOOLS

There are a number of tools available in market for software testing. Some have been used from a very long.time and some new tools have also been developed with a lot of new functionalities. Here, we are going to discuss few tools that are used for automated testing (Nancy Bordelon A comparison of automated software testing tools).

Ranorex

This is a simple, comprehensive and cost effective tool used for automatic testing. It is a better alternative to other testing tools because it tests applications from a user's perspective, using standard language and common programming techniques like C# and VB.net. It does not require understanding a scripting language, because it is coded in pure .net code. Any one of the three languages, VB.net, C# and Iron Python can be used. It is used by a lot of commercial software companies and enterprises around the globe. These simulation tools such can have same problems to the same record and playback methods, as the test plan and test cases are often tightly coupled to the code, and both methods still depend highly on experts to create the correct these tests to ensure full coverage. Future work for ranorex involves creating an easily accessible, open and highly documented interface for the clients to write their own plug-ins, which provides the maximum of recognition for their own applications. Some of the features of this tool are:

- The test automation modules can be created with a standard .NET compiler.
- It provides the ability to do test automation in client's own environment
- It uses standard and modern programming techniques
- It allows testers with little programming knowledge to create professional test plans and cases and modules with Ranorex Recorder
- It does image-based recognition
- It contains Record-Replay functionality which is called Ranorex Recorder
- It provides easy integration for 32 and 64 bit operating systems
- It is built on the .NET Framework
- It offers a standard and flexible test automation interface
- The Ranorex Recorder provides user code actions, which allows developers to provide special validation or automation methods for their testers with less experience in programming
- It targets to get everything flexible and automated
- It supports all the technologies via Ranorex Plug-Ins
- It allows user interface for managing test cases, plans and configurations
- It supports the use of data variables

Rational Functional Tester (RFT)

IBM developed this product in 1999. It is an object-oriented programming based automated testing tool. It includes regression and functional testing tools which note down the results of black box tests in a well scripted format. Once captured, these scripts can be executed against future script builds of any application to verify thatnew functionalities have not disabled any previous functionality. With the help of this tool, black box tests can be run as well as white box tests for code bottlenecks, memory leaks or measuring code coverage. In 2006, IBM made a major transition to its software development platform to better help companies build complex software and applications. The Baltic or IBM Rational 7 was developed in 2006. Some of the advantages of this tool are:

- It enables regression testing
- It frees up Quality Assurance departments from maintaining and executing basic tests plan and cases, and encourages the creation of additional, thorough tests
- It automates other non testing activities such as functional and test lab machine preparation.
- It reduces the probability of human error that can occur during activities such as test step execution and also test result recording

It works with Web based, Java, and Microsoft Visual Studio, .NET, SAP, terminal-based, Siebel and Web 2.0 applications. This product also uses a Object Code Insertion (OCI) technology where no source code is used. This technology looks at the executable files in an application. These tools when built into the software, including Pure Coverage and Purify Quantify, perform white box testing on a third party code. Some of the advantages of these tools are:

- It provides memory leak detection and run-time error
- It records the exact amount of time an application spends in a given block of code for the purpose of finding all inefficient code bottlenecks
- It pinpoints areas of application that have been and have not been executed
- When performing regression tests on a product, if the application changes, like, images in different locations, tests will not fail because the product uses robust

This tool is much similar to others as it enables some users to automate software testing solutions and with the help of this tool it is done in a cloud too. This tool does not require any scripts to be written i.e. only simple English-based tools are used that simplify the task of software implementation with efficient and easy to use tools. Other advantage of this tool is that its cost is very less i.e. $10 per month. There is no such software to download and thus no infrastructural investment is required. Since it is used in the cloud, it has a very quick and easy setup that includes no install. This cloud based software has an easy navigation to home page.

3.1 8 SOFTWARE TESTING TRENDS EVERY TESTER SHOULD FOLLOW IN 2018

oftware testing techniques and methods have changed a lot in the last decade. Inspired by the manufacturing process, software testing has acquired the waterfall approach to test for the quality, which involved step-by-step checks and case tests when the products are at user acceptance phase. This made the task a lot more difficult for manual testers as the amount of data to be processed to run all the test cases was too much.

The emergence of Agile and DevOps methodology by many business organizations insert the testing and checking process at an earlier phase of the process. Similarly, there had been considerable changes in Software Testing methods in past that needs to look forward in future. So let's check out the following testing trends which will transform the future prediction:

1. Shift from QA to Quality Engineering

The world is changing continuously, and every now and then, there is a talk about new technology surfacing in the market. Quality Assurance (QA) follows a systematic waterfall approach for testing which is a step by step process, making it thorough but lengthy at the same time. Hence, QA is struggling to keep up with the changing dynamics in the testing field. QA can sometimes perform as a bottleneck to a complete flow of processes. As it follows a step-by-step test, the previous process needs to be completely done for the next to begin and beings a manual tester one can only look after this in detail. But with QA the amount of data and processes can easily pile up. With Quality Engineering one can introduce the testing and automation process earlier in the process rather than at the customer's acceptance phase.

2. IoT

IoT (Internet of Things) is one of the fastest growing technologies in today's world and IoT is challenge for Test Automation. A complete web of things is inter-connected with each other through the internet (still sounds like one of those sci-fi movies). The hardware is controlled by dedicated software which connects them to the internet and from there it connects to all other things. As great as it may sound, there are a number of vulnerabilities in the system. Hence, the products which are connected should be tested for functionality, quality, and most important security. As per HP reports, around 70 % of the systems connected to IoT are vulnerable to security-related issues.

3. Digital Transformation with Agile

Agile Methodology has been used for the last 17 years after it was discussed in Manifesto of Agile Software Development. This methodology promotes working with various small teams in collaboration and while doing these tasks & taking on a smaller amount of processes and makes a quick and continuous delivery. Modern practices based on Agile Manifesto can be really helpful with User Experience (UX) work like planning, execution, and evaluation.

4. DevOps

DevOps is a term used for a particular set of rules or principles to reduce the amount of time from development to operations. DevOps is not particularly a new concept in business but its emergence in the technical field is quite recent and in the past 5 years, it has gained a tremendous amount of support from the business organization. In the coming years, more organizations are likely to adapt to these set of principles to improve their overall performances as its emphasis on Automation and Integration.

5. Time for Big Data Testing

We live in the golden age of technology, where the clients and users on various platforms upload terabytes of data and so managing such amount of data, it needs a unique approach for testing. Big Data Testing is a process which can be helpful for business to test such large amounts of data. The main aim is to test the data for quality to start with. Big data is a really large amount of datasets which can't be processed by traditional computing techniques.

6. Bigger Market share for Mobile Users & Test Automation

Mobile is now one of the most important parts of an individuals' life. Nowadays there is an app for everything where the number of devices, OS platforms and software runs on single app. Testing mobile applications is a much more complex task than testing websites and the number of updates, types of devices and software updates coming every day makes it even harder to keep up with. Hence cost and market readiness are really important in future, Mobile app testing automation can be really vital.

7. API and Micro Services Test Automation

Microservice is basically a method of developing software to specifically test for any particular conditions. Types of services can be generated and by establishing a connection between them, the whole task can be divided into parts. In this, each service is created in such a way that it can perform a particular set of processes. This gives you the freedom to make changes in any smaller specific area of the application which requires changes instead of changing the whole system.

8. Increasing Adoption of Open Source Tools

Open source tools are really beneficial for business and are going to play a vital role in future too. There are many advantages of using Open Source Tools other than the cost, as it is a free to use and available to the public. It can be easily customizable, is more flexible than some expensive proprietary stuff and it is open for public. Users do have a hand in designing so it really gives

you the freedom to design the way you want and there are many integrations for your powerful Test Automation too. A debatable point could be that of security, as being available to the public is not actually a definition of secure but when it goes through more sets of eyes the chances of finding out the bug and fixing it increases.

This article talks more about the methods and principles of Software Testing, which can be really important in changing and speeding the process. This will result in faster updates, better user experience and further advancement to the next level of computing.

conclusoion

Testing is a critically important verification method that takes up a very large portion of a project's resources, including schedule, budget, staffing, and facilities. Unlike the many constructive activities of systems engineering, testing is relatively unique because it is inherently destructive. Its primary purpose is to force the system or its components to fail so that the defects that caused the failure can be uncovered and then fixed. In addition to defect detection, testing is also performed to provide sufficient objective evidence to justify confidence in the system's quality, fitness for purpose, and readiness for being accepted and placed into operation.

REFERENCES

[1] Debugging available at http://en.wikipedia.org/wiki/Debugging

[2] Software deployment available at http://en.wikipedia.org/wiki/Software_deployment

[3] Software maintenance available at http://en.wikipedia.org/wiki/Software_maintenance

[4] Software development process available at http://en.wikipedia.org/wiki/Software_development_pr ocess

[5] Involve testing throughout the SDLC available at http://www.silverpath.com

[6] IEEE(1990), IEEE Standard Glossary of Software Engineering Terminology , Los Alamitos, CA: IEEE Computer Society Press.

[7] Zhang Hongchun, Research on New Techniques and Development Trend of Software Testing.

[8] Nancy Bordelon, A comparison of automated software testing tools

[9] **https://www.softwaretestinghelp.com/types-of-software-testing/**

[10] **http://ijcsi.org/papers/IJCSI-11-2-2-120-123.pdf**

[11] **https://www.computer.org/csdl/mags/co/2014/02/mco2014020021.pdf**

[12] http://inpressco.com/wp-content/uploads/2014/07/Paper122368-2372.pdf

[13] https://www.ugc.ac.in/journallist/ugc_admin_journal_report.aspx?eid=MTY0

[14] https://www.researchgate.net/publication/312484469_Software_Testing_Techniques_A_Literature_Review

[15] https://www.testing-whiz.com/blog/8-software-testing-trends-every-tester-should-follow-in-2018