THE IMPACT OF TEST STRATEGY IN SOFTWARE TESTING

UNAOWO MONDAY BRANDY

***Abstract*— Software projects today are more gigantic than ever before with teams spanning the globe working on the same program test strategies are changing rapidly. A culture of quality in all part of the organization is essential for software testing successes.**
This paper is concerned with diagnosing the impart of test strategy in software testing through analyzing the factors considered before software testing effort is imitiated software testing must be planned carefully to avoid wasting development time and resources. Test specification document should be reviewed like all other software engineering work products.

***Index Terms*— strategy, test, software testing, test strategy.**

I. INTRODUCTION

According Jeff Nyman (2011), a test strategy is a framework for making decision about value. It is a set of choice and a set of idea. The choices and ideas defines the activity that you will undertake as you test. Test strategy should at first address two important factors

- Risk that matters most
- Someone's interest

A test strategy is the connection between your testing activities and the purpose of your testing function. It is what you do in a given situation to accomplish the purpose of the testing. The choice of test strategy is one of the most powerful factor in the success of the test effort and the accuracy of the test plans and estimates. According to John Overbaugh (2014), a test strategy is a loslispic view to how you will test a product. It is the approach you will take, the tools and methodologies you will use to deliver the highest possible quality, the test strategy consists of a myriad of methodologies, activities, and staffing solutions. The test strategy sets the overall acceptable bar and calls out how the team will archieve that bar. It is tha sum of all the inputs, in an organized plan.

In the order hand, software testing is a set of activities that can be planned in advanced and conducted sequentially. For this purpose a template for software testing are developed to define a set of steps into which we can place specific test case design techniques and testing methods for the software process. A member of software testing strategies have been proposed in the software testing literature in the past. All provide the software developer with a template for testing and all have the following characteristics.

Manuscript received May 21, 2015

UNAOWO MONDAY BRANDY, Post graduate student (M.sc information technology) SRM University, kattankulathur-603203, Chennai, India

Testing begins at the component level and works outward toward the integration of the entire computer-based system.

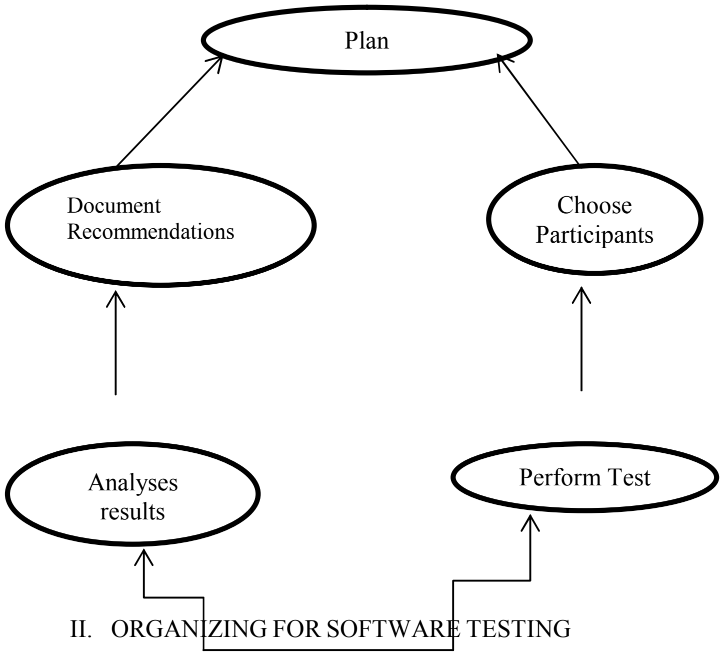
Different testing techniques are appropviate at different point in time.

Testing is conducted by the developer of the software and by independent test group for large projects.

Testing and debugging are different activities, but debugging must be accommodated in any test strategy.

A strategy for software testing testing should accommodate low-level test that are necessary to verify that a small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements. A strategy must provide guidance for the practitioner. Since the steps of the test strategy occur at a time when when deadline pressure begins to rise, progress should be measurable and problem should surface as early as possible. Software testing is a process that can be systematically planned and specified. Test case design can be conducted, a strategy can be defined, and results can be evaluated against prescribed expectations.

Fig1: Generic software test strategy model



II. ORGANIZING FOR SOFTWARE TESTING

The software engineer create a computer program, its documentation, and related data structures. Like any constructive builder, the software engineer is proud of the edifice that has been built and looks askance at anyone who attempts to tear it apart when software testing commences, there is a subtle yet definite attempt to 'destroy' the thing that

the software engineer has built. Psychologically speaking, software testing can be considered as distractive.
The software developer is always responsible for testing the individual units of the program, ensuring that each performs the functions for which it was designed. In many cases, the developer also conducts integration testing.
A testing step that leads to the construction and test of the completed program structure. It is only after the software architecture is completed that an independent test group becomes involved.
The role of an independent test group is to remove the inherent problem associated with letting the builder test the thing that has been built. Independent testing helps to removes the conflict of interest that may otherwise be present. She is part of the software development project team in the sense that it becomes involved during the specification activity and stays involved throughout a large project6. The independent test group reports to the software quality assurance organization, thereby gaining independent that would not otherwise be possible.

III. A SOFTWARE TESTING STRATEGY

The software engineering process may be viewed as the spiral illustrated in fig.1 below. Initially, system engineers define the role of software that leads to software requirements analysis.
Where the information domain, function, behavior, performance, constraints, and validation entwine for software are established.
Further in ward along the spiral we come to design and finally to coding. To develop computer software, we spiral in ward along streamlines that deer eases the level of abstraction on each turn.
A strategy for software testing may also be viewed in the context of the spiral (fig-1). Unit testing begin at the vortex of the spiral and concentrates on each couponing of the software as implemented outward along the spiral to integration testing, where the focus in on design and the construction of the software architecture. For another turn outward on the spiral, we come across validation testing, where requirements analysis are validated against the software that has been constructed. Finally, we take a turn that arrives at system testing where the software and other elements are tested as a whole. To test computer software, we spiral out along stream lines that broaden the scope of testing with each turn.

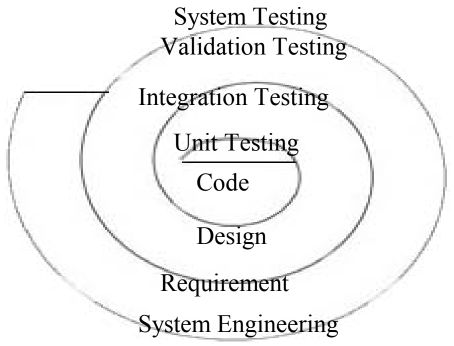


Fig.2 Spiral Testing Strategy Model

IV. TYPES OF TEST STRATEGIES

The major and commonly used test strategies are gunned into the following types:

- Dynamic test strategy: Dynamic strategies, such as exploratory testing is concentrating on finding as many defeats as possible dunning test execution and adapting to the realities of the system under test as it is when delivered. For example, you may create a lightweight set of testing guidelines that focuses on rapid adaption or known weaknesses in software.
- Standard-compliant test strategy: This is about reliance upon an externally developed approach to testing with little customization and varying time involvement in testing. For example, time involvement in testing. For example, one may adopt the IEEE 8z9 standard for testing, using books such as [craig,2002] on [Drawback, 2004] to fill in the metrological gaps one may also adopt one of the agile metrological such as extreme programming.
- Regression-averse test strategy: This is a set of procedures that is usually automated and detest regression defeats. It moles automating functional test prior to release of the function. It focuses on testing functions that already have been released which form post release effort.
- Consultative test strategy: this have in common the reliance on a group of non-testers to guide and perform the testing effort with emphasizes on later stages of testing simply due to the lack of recognition of the value of early testing. For example, you may ask the user or developer of the system to tell you what to test or even rely upon them.
- Methodical test strategy; this strategy; this strategy have in common the adherence to a pre-planned, systematized approach that has been developed internally, assembled from various concepts developed in-house and adapted from outside ideas at various feasting time efforts. For example you designing and poetizing the test based on risk. Another analytical test strategy in the requirements based strategy, where an analysis of the requirement specification forms the basis for planning, estimating and designing test.

Half of these strategies are more preventive whereas others are more reactive. For example analytical test strategies involve upfront analysis of the test basis and tend to identity problems prior to test execution. This allows the early and cheap removal of defeats' which forms the strength of preventive approaches.
On the other hand Dynamic test strategy focuses on the test execution period that allows the location of defeats and defects clusters that might have been hard to remove until the actual system is developed. This forms the strength of reactive approaches.
Rather than see the echoic of strategies (preventive or enactive) , it in better to understand that there is no one best way to test a particular situation, therefore feel free to bonnet and blend May have a checklist that you put together over the years that suggest areas of testing to run on you may follows industrial standard for software quality such as ISO 1926 to test your major outlined areas.
Model-based test strategy: this type of test strategy is about the selection of some formal or informal model for critical system behaviors during the requirement and design stage of the project for example, you can build a mathematical model for loading and responding to e-commerce servers and test

based on the model. The system is deemed fit if the behavior conforms to that predicted by the model.
Analytical test strategy: This involves the use of formal or informal analytical techniques usually during the requirement and the design stage of the project. For example, the risk-based strategy involves performing a risk analysis using project document and stakeholders input, then planning, estimating, impact of test strategic. The impact of test strategies in software testing in better understood in considering the following factors that are of vital importance in software testing.
Risk: Risk management is very critical during testing; we have to consider risks and the level of risk. For a defined application that is metamorphosing slowly, regression is an important risk strategy that makes sense. For a brand view application, a risk analysis may save the trouble and so we go for risk-based analytical strategy.
Objectives: software testing as a matter of fact must satisfy the needs and requirements of the stakeholders to be successful. If the objective in to find as many defats as possible with a minimal amount of disposable time and effort invested, just like an independent test laboratory, a dynamic strategy would be comfortable.
Skills: A strategy is not only chosen, it must be executed. A standard compliant strategy is a good choice when you lack the time and skills in your team to create your own approach.
Product: Product like weapon systems and contract-developments. This synergies with a requirement-based analytical strategy.
Regulation: In addition to stakeholders, regulators should satisfied in an test, efforts. In this case, we may need to plan a methodical test strategy that satieties these regulators that we have met all their requirements.
Business: Business considerations and business continuity are often important. So if we are to use a legacy system as a model for a view system, then a model based strategy is a good option.
We must close testing strategies with an edge towards the factors mentioned earlier, the schedule, budget, feature constraints of the project, the realities of the organization and its politics must mention here without any fear of contradiction, that a good team with wise test strongly can sometimes triumph over a situation where materials, process and delaying factors are ranged against its success. However, talented execution of an unwise test strategy is like going very fast down a highway in the wrong direction. Therefore, we must make smart choices in terms of software testing strategies.
Debugging strategies and bug removal consideration debugging is the process of removal of a defect which occurs as a consequence of successful testing debugging is not testing but always occurs as a consequence of testing (fig.3). Some people are better at debugging that others. The debugging process has two basic outcomes.

1. The cause will be found and corrected or
2. The cause will not be found

The debugging process attempts to match symptom with cause there by leading to error correction. Correction approaches of debugging are brute force, backtracking and cause elimination.
Bug removal consideration may include:

- Is the cause of the bug reproduced in another part of the program?
- What "next bug" might be introduced by the fix that is being proposed?
- What could have been alone to prevent this bug in the first place?

CONCLUSION

There are lots of flavors in software testing. The test strategy in the summation of all the inputs in an organized test plan. The impact of test strategy in software testing is to produce a light probability of finding errors in software testing process with a minimum of resources, skills efforts and time. The software testing process is not redundant, it is neither too simple not too complex. It is a template for testing.

ACKNOWLEDGMENT

The author salutes the financial assistance of elder (surv) kufe ufot and Dr. Prince lnyang inyang which enable him to make registration for this publication. He also remains grateful to S.kanchana for her effort in grooming him in software process course.

REFERENCES

[1]Mark A. Garzone (2014), software testing: A guide to testing mobile Apps, website and games. Create space independent publishing platform.
[2]James Bach (2013), Heuristic test strategy model. Adapted from: http://www.satisfice.com/tool/htsm.pdf
[3]Jeff Nyman (2011), strategy and test plan. Adapted from: hhtp://www.testerstories.comm/2022/22/your-teststrategy
[4]Brain Hambling et al (2009), software testing: An iseb intermediate certificate illustrated. BCS publishers.
[5]John over Baugh (2007), testing methodologies, testing strategies and testing types. Adapted from: hhp://www.cloud.continuedreadig
[6]R.S pressman (2005), software engineering: A practical approach. 6/e. R.S. pressman and associates, inc
[7]Watts S. Humfrey (1995), A discipline for software engineering. Addison-wesley professional publication.
[8]Ricck D.Craiy (2002), systematic software testing artech house publishers
[9]Can kaner et al (2002), lessons learned in software testing: A context-Driven approach (computer science). John willey and sons publication
[10]Cem kaner et al (2001), software testing! A context driven approach wiley publication
[11]Martin pol et al (2001), guide to the Tmap approach Addison-wesley publication. iSEN 0201745712
[12]httpp://www.istqbexaucertifiacation/com/what
[13]httpps://www. atlassian.com/softwaretesting
[14]httpp://www.en.m.wikipedia.org/wiki/test-strategy.

AUTHOR

The author "Mr Unaowo Monday Brandy" is currently a post graduate student (M.Sc. Information Technology) at SRM University, Chennai. He had obtained B. Eng. (Computer Engineering) from the University of Uyo in 2010.
The author is an advocate of the millennium development Goals (MDGs). He has won statement award and outstanding corp member of the year in Anambra state, Nigeria. He is a

graduate member of the Nigeria Society of Engineers. (NSE). He has publish papers in journals including “Virtualization Head Roads in Information Technology” in International Journal of Computer Trend and Technology (IJCTT).