Comparative Study on Software Testing Models

K.Sivakami[1], Dr. K.Mohan Kumar[2]

[1]Research scholar, CS Department, Rajah Serfoji Govt. College, Thanjavur, India
[2]Asst.Professor and Head, CS Department, Rajah Serfoji Govt. College, Thanjavur, India

ABSTRACT: There are many products and services such as bank services, health care services, retail business and other electro-items like theft alarm, micro ovens, and mobile phones in our daily use. Software is major commodity of these products and services. The demand of software quality is increasing as the software is becoming more and more important to us. This leads to considerable challenges for Information Technology industry in testing practice. The practical challenges may include human resources, hardware availability, software/application readiness, inadequate testing timeline and tight software development schedule. Software testing strategies are very important to test the functionality of the software without compromising on its quality. Any strategy adopted for the software development and deployment are expected to test functional behavior, logical data dependencies and integral behavior of the Software. The role of defining strategy is to detect possible defects in system and can help in successful completion of the system according to functionality.

KEYWORDS: Waterfall, Risk Based Test Model, Iterative Test Model, Desktop Integration Test Model, JAWS, Test Effort Estimation, Function Point.

I. INTRODUCTION:

With the passage of time, software products are growing larger and becoming more and more complex. It leads to more opportunities for defects to sneak in software development and its maintenance. Software takes lots of resources of software development organization and they are interested to provide software in functional form as long time as possible. Due to changes in the environment and in the user requirements, it is very important that the software is easy to adapt and maintainable. This leads to challenges in the testing practice now a day. Waterfall test model is the traditional strategy that is being adopted for the testing in the software development.[1]

II. TESTING OVERVIEW:

Definition: The formal definition of testing was introduced in 1979 by Myers as "the process of executing a program with the intent of finding errors". According to this definition, fault detection is primary goal. Myer's goal was to show that program has no faults; one might select the test data which has low probability of finding errors. If the goal is to find errors, one will select test data which have high probability of detecting errors and our testing succeeded successfully.[2]

Testing Techniques: The guideline published in the Institute for Computer Sciences and Technology of the National Bureau of Standards in 1983, introduced a methodology which integrates analysis, review and test activities to provide product evaluation during the software life-cycle. The two key testing techniques recommended widely are Black-Box testing and White-Box testing which is detailed further below in this section.

Black-Box testing - is testing software based on output requirements and without any knowledge of the internal structure or coding in the software.

White box testing - is highly effective in detecting and resolving problems, because bugs can often be found before they cause trouble.[3]

III. TRADITIONAL TEST MODEL - WATERFALL:

The activities that comprise the creation of software are commonly modeled as a software development lifecycle (SDLC). The software development lifecycle begins with the identification of requirements for software and ends with the formal verification of the developed software against those requirements. The software development lifecycle does not exist by itself; it is in fact part of an overall product lifecycle. Within the product lifecycle, software will undergo maintenance to correct errors and to comply with changes to requirements. One of the more commonly accepted lifecycle models is the waterfall model, also known as the linear sequential model. The waterfall model, while still popular, is not conducive to the rapid development cycles that the Internet atmosphere demands, today's strict economics, and increased expectations of quality. Specifically, following are the particular concerns:

- Typically, more time than was initially scheduled is needed to integrate subsystems into a complete, working application. This squeezes the time allocated to testing, always the first item to be cut. The quality of software is poor.
- Design flaws that require significant changes to the product are discovered late in the software cycle. Rarely is tangible design validation performed in the project's early stages. And the releases are long enough where the

customer/users have to wait for considerable amount of time. Customer satisfaction is negatively impacted in the waterfall test strategy.

- Today, customers use various types of hardware, operating systems and other utilities/application with which the software has to function in order to be business as usual. Waterfall method does not conduct such test and hence, the software fails after production release due to the hardware/software conflicts.
- Waterfall model do not support certification of software for visually impaired users.

IV. RECOMMENDED TEST MODELS:

The aim of this analysis is to propose an appropriate test model for projects that fall into the categories of the following testing challenges:

1. Timeline/Resource limitation.
2. Customers satisfaction/Less wait time for project releases
3. High volume of production failures due to hardware/software compatibilities.
4. Software does not comply for visually impaired users.

The objectives which are formed in this analysis to solution the above mentioned challenges are:

- Identify a robust test model for a successful software development without compromising the business need of the software.
- Identify an effective test model for multiple releases of software to meet the customer satisfaction of availing new business functions to their clients.
- Identify a robust test model in order to certify the software is compatible with major varieties of hardware and software.
- Identify a best automated tool to certify the software is complying for the visually impaired users.

V. RISK BASED TEST MODEL (RBTM):

Since it's impossible to effectively test every part of the every application, organizations struggle to decide how much testing to do on which application modules, and in the proper sequence, to meet customer needs at the lowest possible cost. Failing to prioritize these efforts properly results in the following outcomes:

- Not finding critical defects until late in the development cycle, which significantly increases the time, and cost, required to fix them.
- Wasting time and money testing less important parts of the application that contain fewer likely defects.
- Higher software testing and remediation costs, by testing a large number of test cases than are necessary.
- Delays in delivering software to internal or external customers.

A far more effective approach is Risk Based Test Model, which examines at the level of the business scenarios the likelihood of a given defect and business impact of such defect. This helps assure the companies spend the most time and money on the most critical areas, and find the largest number of defects (and the most important defects) in the shortest time and at the lowest possible cost.

RBTM is an approach recommended for the projects with schedule limitation. This model is carried out with manual effort of testing and focuses on the functional attribute of the software.[4]

Approach: RBTM uses the approach of prioritizing what software to test based on:

- Risk of the failure of a given software module or that is not tested.
- The impact on the business of such failure.

Based on this analysis, project managers can focus testing on the most important scenarios or cases, selecting the amount of testing to do based on their project constraints and the amount of risk the company can afford to take.

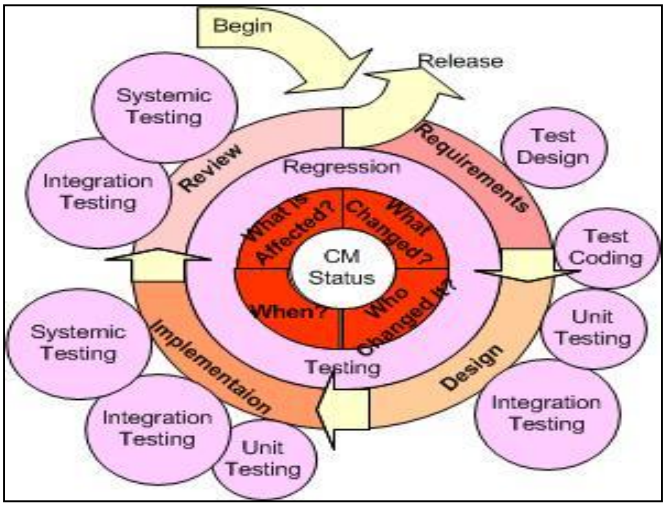
RBTM based testing is a strategy of software testing in which the features and functions to be tested are categorized by priority, importance and potential impact of failures. Using risk, Risk based testing prioritize and emphasize the suitable tests at the time of test execution. In other word, Risk is the chance of event of an unwanted outcome. This unwanted outcome is also related with an impact. Some time it is difficult to test all functionality of the application or it might not be possible. Use Risk based testing in that case; it tests the functionality which has the highest impact and probability of failure.[5]

A properly implemented RBTM methodology helps assure that high-risk areas are tested first, then medium-risk and, finally, low-risk areas. This is an improvement over non-prioritized testing, in which low-risk areas might be tested before higher-risk modules.

Risk-based testingis the process to understand testing efforts in a way that reduces the remaining level of product risk when the system is developed. [6]

VI. ITERATIVE TEST MODEL (ITM):

One recent approach that appears to have sound engineering principles, good economics, and consensus in the community is the iterative software lifecycle. The iterative lifecycle has the benefit of application of lessons learned in the waterfall model.



For each cycle of the model, a decision has to be made as to whether the software produced by the cycle will be discarded, or kept as a starting point for the next cycle. This approach has been referred to as incremental development. Iterative Test Model is the only appropriate test model for the software validation that is been developed using agile method.

It is apparent that testing activities are committed throughout the lifecycle. It is very likely that other lifecycle models could support similar techniques, as any limitations are more psychological than technical. However, the deep-rooted culture of design, build, and then hand-off over the wall for testing is entrenched in the minds of a great majority of project stakeholders, managers, and developers. The iterative model attempts to implement total quality engineering horizontally across the project, with everyone involved.[7]

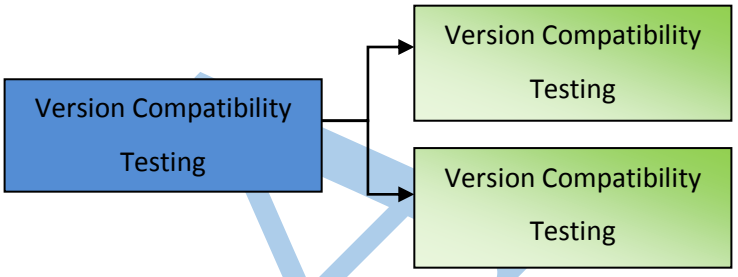
VII. DESKTOP COMPATIBILITY TEST MODEL (DCTM)

Compatibility is nothing but capability of existing or living together. In normal life, oil is not compatible with water, but milk can be easily combined with water.

In computer world, compatibility is to validate whether particular software is capable of running on different hardware, operating systems, applications, network environments or mobile devices. Compatibility testing is a type of the non-functional testing that includes the following test types:

- **Hardware –** It validates the software to be compatible with different hardware configurations. The hardware configuration varies from:
  - Type of computer – Desktop, Laptop, Palmtops, Table PC, etc.,
  - Make & Model of computers – Dell-X45, HP, IBM-Lenovo, etc.,
  - Configurations – processor speed, RAM, etc.,
- **Operating Systems –** It validates the software to be compatible with different operating systems line Windows – XP, Vista, Unix, Mac OS, etc.,
- **Software –** It validates your developed software to be compatible with other software. Examples
  - MS Word, Excel, Outlook, VBA, Java Runtime, JVM, etc.,
- **Network –** It evaluates the software's performance in network with varying parameters such as Bandwidth, Operating speed, Capacity. It also checks application in different networks with all parameters mentioned earlier.
- **Browser –** It validates the software to be compatibility of the web application with different browsers like Firefox, Google Chrome, Internet Explorer, etc.,
- **Devices –** It validates the software to be compatibility of the software with different devices like USB port devices, Printers, Keyboards, Mouse and Scanners, other media devices and Bluetooth.
- **Mobile –** It validates the software to be compatible with mobile platforms line Android, Windows, iOS, etc.,
- **Versions –** It validates the software to be compatible with other system utilities like service pack SP1, SP2, SP3, etc.,

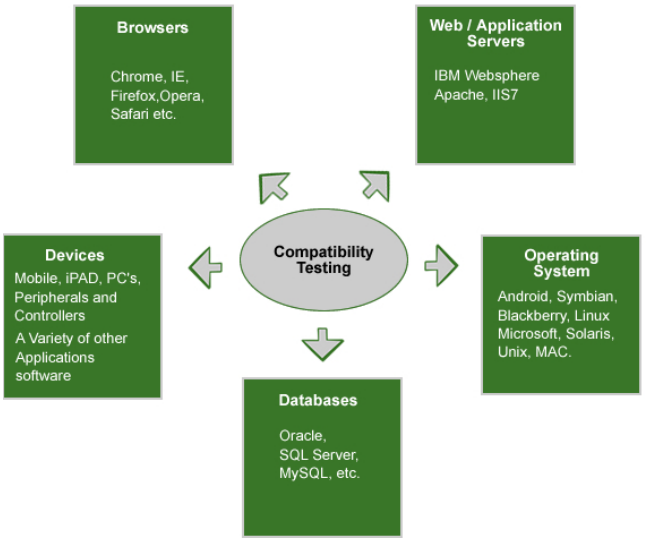
Following are two types of version validation:



Backward Compatibility Testing is to verify the behavior of the developed software with the older version of the software.

Forward Compatibility Testing is to verify the behavior of the developed software with the newer version of the software.

- **Installation –** Installation testing is performed once the application functional testing is complete. As part of the installation test, installation of the application executable on the previously installed application while the application is in open and closed status. Validate that the application continues to work without any negative impact to the previous version along with the newly modified changes.
- **User Experience –** Finally, the executable is wrapped with the packaging skin. The text, informative messages, logo and other details are validated for better user experience.



Approach of Desktop Compatibility Testing is as follows:

Assessment – Evaluate the software that is planned to undergo changes/updates for technical dependencies, example; Let us assume that a software called "Retirement Planner" is dependent on the PDF application. Identifying all such dependencies takes place in this phase and a complete list is prepared.

A matrix is constructed containing the information about the upstream, downstream and prerequisites of the software that is planned to undergo a change.

Considering the above two evaluation, the impact analysis report has to be prepared that will list all the software and its impact on itself and other applications.

The impact report becomes the requirements and that are communicated to the testing team and any other cross commit teams.

Remediation – Solution is identified considering the direct and indirect dependencies of the other software or utilities and the impact analysis report generated in the Assessment phase.

Once the solution is accepted by the steak holders of the project and other business heads, the required code changes are made and detailed unit testing is done on any one or two specified hardware configuration.

Once the unit testing is complete, the developer has to communicate to kick-off the testing effort.

During all these effort of development team, the testing team has to prepare matrix on the hardware, devices, networks that are considered for testing and the same has to be aligned with the business team in order to broadcast the risks of for any configuration not be considered for testing.

Note: It is testing team's responsibility to identify the entire possible configuration for testing based on the usage of the configurations in the client field.

Testing –Testing should kick-off the testing once the test software is deployed on the business aligned test machine configurations. Testing team has to conduct the following types of testing:

- Functional/system test – testing the modules in the software that under goes changes for its functionality.
- Regression test – testing the entire software for all of its key functionalities for any negative impact of the new changes made in the software. This could be performed either by the technical testing team or the business testing team.

In order for a quality regression test, the tester should have a very good knowledge about the software from the business perspective and hence, the business team performing this test is advisable.

If the technical team has to perform the regression test then the screen comparison from the previous releases is the recommended approach.

- Compatibility test – testing the other software in concurrent usage of the software that is being modified. Issues/defects due to concurrent usage of the software, utilities and application have to be identified as part the integration testing.
- Testing can be done by manually/automatically comparing the screenshots or the actual test results of previous test runs.
- On unsuccessful completion of the testing, the issues has to be raised as defects in the defect management tool and testing team has to arrange for defect triage meetings inviting the development team, business team, project sponsors and others if any.
- Depending upon the timeline and criticality of the project, the defect triage may be conducted once or twice a day. Testing is responsible to host the triage and the all other communication, defect tracking and etc.,
- On successful completion of the testing, the sign-off from technical end has to be communicated to all the stake holders.

Implementation – Once the testing is complete, the software will be rolled out to the clients through the software deployment tools like Altiris.

Environment for performing the desktop compatibility testing is the labs where the physical computers, devices, printers, scanners, keyboards, mouse are available. Also, the other software and the software to be verified are accessible from that lab through CD format or from share points or through deployment tools like "Altiris".

A lab administrator is necessary to maintain and secure the lab. In addition, the following services have to be provided to conduct the compatibility test:

- Building the machines with all the required software, utilities, network connections, printer configurations, etc.
- Technical support in fixing the issues that may occur in hardware, network, etc while testing
- Coordinating with cross commit teams that may be required for test.

TESTING TOOL TO CERTIFY SOFTWARE FOR VISUALLYIMPAIREDUSERS:

Freedom Scientific is an organization that develops the highest quality video magnifiers, Braille displays, screen magnification software, and the #1 screen reader, JAWS® for Windows. For over 20 years, our products have provided access to print and computers for people with blindness, low vision, or learning disabilities.

It is important to evaluate the accessibility of web content with a screen reader, but screen readers can be very complicated programs for the occasional user, so many people avoid them. This doesn't need to be the case. While screen readers are complicated, it is possible to test web content for accessibility without being a "power user." It is a powerful software program designed to work with a speech synthesizer to improve the productivity level of visually impaired employees, students and the casual user. By streamlining keyboard functions, automating commands, and eliminating repetition, JAWS allows the operator to learn faster and easier than ever before. JAWS is based upon a whole new approach to talking

computers - that of designing software with the priorities of the blind user in mind. Yet, the sighted trainer or supervisor has not been forgotten, since JAWS offers both audible and visual flexibility.[8]

COMPARISON OF TESTING EFFORT ESTIMATION BETWEEN WATAERFALL, RBTM AND ITM:

The objective of this analysis is to describe the effective test models to help meet the challenges of testing practice. Importantly, the traditional approach needs to be revisited, and all parts of an approach that do not help or not applicable need to be thrown overboard. The approaches discussed in the analysis are not standardized models but has to be customized as required to the context of the software development and its business objective. The recommended test models are compared with one another and with the waterfall method by the parameter "Test Execution Effort". Similarly, JAWS is compared with other screen reading tools available in the market by the parameter "Utilization". This utilization numbers are taken from the reference mentioned in "References" section.

The table below is a sample data of 11 test cases with risk selection made whether or not the test case has to be tested for the current software release. The risk is purely identified by the subject matter expert, the business analyst who can judge on the functions criticality and severity in order to identify the test cases for testing.

System: My Bank Accounts

System Feature	Test Case #	#. Of steps	TC Category	TC Weightage	TC Points	FP Weightage	Priority	Risk Selection	FP Weightage
Security Module	Test Case 1	10	Medium	3	5	15	High	√	15
Security Module	Test Case 2	5	Simple	1	3	3	High	√	3
Authentication Module	Test Case 3	3	Simple	1	2	2	Medium	√	2
Authentication Module	Test Case 4	3	Simple	1	2	2	Low	×	-
My Accounts Page	Test Case 5	15	Complex	5	10	50	High	√	50
Accounts - Nickname	Test Case 6	5	Simple	1	2	2	Low	×	-
Accounts - Address Update	Test Case 7	5	Simple	1	3	3	Low	×	-
Accounts - Pin Update	Test Case 8	3	Simple	1	1	1	Medium	×	-
Accounts - Linking	Test Case 9	10	Medium	3	7	21	Medium	×	-
Money Transfer – Internal	Test Case 10	10	Medium	3	4	12	Medium	×	-
Money Transfer – External	Test Case 11	10	Medium	3	8	24	High	√	24

Let us calculate the effort using the Function Point effort estimation technique. In this FP technique, each functional point should be given weightage and the test cases have to be categorized into three categories as "Simple", "Medium" and "Complex". So, the total effort estimate is calculated by the multiplication of Total Function Points and Estimate defined per Functional Point.

Total Effort Estimate = Total Function Points * Estimate defined per Functional Point

Effort is calculated for all FPs in waterfall methods and it is as follows.

Total Effort Estimate = 135 x 5 = 675

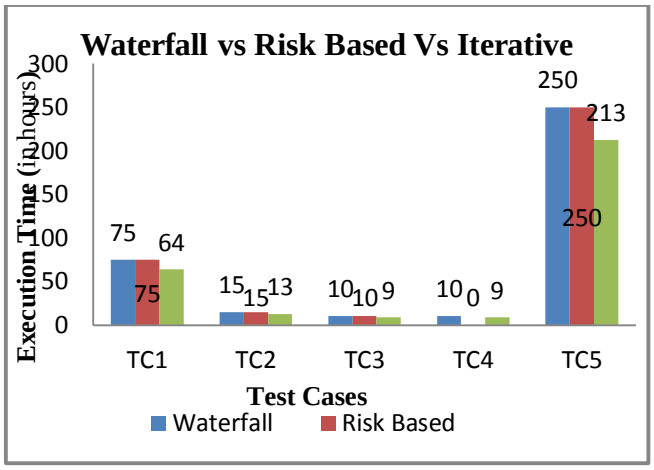
Effort is calculated for the test cases under risk in the RBTM and it is as follows:

Total Effort Estimate = 94 x 5 = 470

Note: The defined effort per FP is assumed to be 5 minutes

The test effort difference between waterfall, risk based and iterative test methods is shown in the below table and chart.

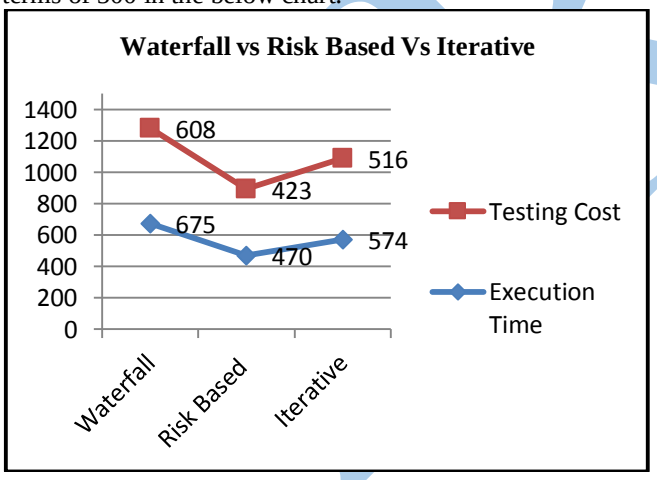
Test Case #	Waterfall	Risk Based	Iterative
TC1	75	75	64
TC2	15	15	13
TC3	10	10	9
TC4	10	-	9
TC5	250	250	213



The below table shows the comparison of the execution time and cost between Waterfall, Risk Based and Iterative testing methods:

Time/Cost	Execution Time	Testing Cost
Waterfall	675	608
Risk Based	470	423
Iterative	574	516

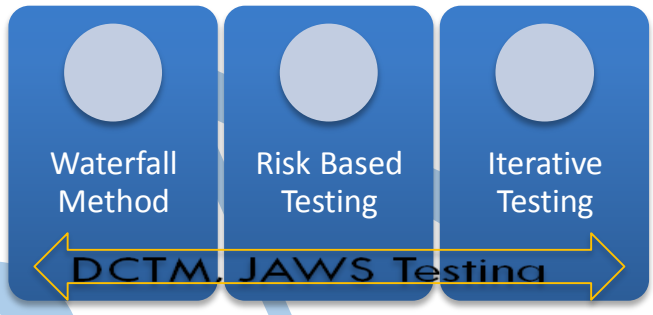
Cost is calculated as Execution Time x 450 and the comparison of cost between the methods is represented in terms of 500 in the below chart.



JAWS is still most popular screen reader tool, it has seen a significant decline in the primary usage – down to 49% from 66.4% in the previous years 2009 to 2012. Other screen reader tools like Window-Eyes and Zoom saw small increases in primary screen reader usage while Voice Over saw small decrease in usage. NVDA saw continued increase in usage, up to 13% from 29% in 2009 and 8.6% in 2010 (nearly 500% increase in just 2.5 years). JAWS is much more popular in Asia (68% of respondents), Australia (58%), north America (50%) than in Europe/UK (37%).

CONCLUSION:

This analysis has recommended four test strategies which effectively address the schedule, budget constraints and interim releases. Among the four test models discussed two test models DCTM and JAWS are horizontal testing conducted across any test model adopted for functional test of the software. The same is depicted in the picture below:



REFERENCES

[1]. Laurie Williams, testing Overview and black Box testing Techniques, 2006.
[2]. Karl Reed, Software Reliability, Testing and Security Class Lecture Notes, computer Science department, La Trobe University, Australia, 2012.
[3]. Sahil Batra and Dr. Rahul, "IMPROVING QUALITY USING TESTING STRATEGIES" Journal of Global Research in Computer Science, Volume 2,No.6,June 2011.
[4]. Isha Int. Journal of Engineering Research and Applications, April 2014.
[5]. Www.intensetesting.wordpress.com
[6]. http://www.softwaretestingclass.com/what-is-risk-based-testing-in-software-testing/#sthash.muWzpLKW.dpuf
[7]. http://home.c2i.net/schaefer/testing.html
[8]. http://www.freedomscientific.com/Products/Blindness/JAWS

