Global Journal of Computer Science and Technology
Software & Data Engineering
Volume 12 Issue 12 Version 1.0 Year 2012
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Approach to Quality Testing

By Drakshaveni G
MCA, BMSIT, Bangalore

Abstract - Time is the most important resource for any activity. Software development is no exception. With an increase in the competition in the market, it is very essential for the companies to release their products into the market at the earliest with good quality to earn profit. In order to develop the products early, companies implement various techniques like Rapid Application development and implement Agile methodologies like Scrum & Xtreme Programming to obtain tangible and useful features of the software at the earliest. It is just not sufficient to develop the product, it is more important to develop a quality product. But how does one know if the product is of good quality or not? This question is best answered by the Quality Assurance team which Tests the product end to end against requirements and standards. They conduct various kinds of tests and check the behavior of the system/ product in various conditions. If it passes all kinds of tests (which is dependent on the kind of the product or the system), then the QA team assures that the product is fit for use. Hence it is very important for the QA time to be given enough time to test the product/system before it is released into market. But it is difficult to judge as to how much time is required to test a product / system completely. An ideal answer would be "Years". Time is a major constraint in any software activity. Many software projects are time bound. Then how does one ensure that within a given time, the product/system is tested to the level where a confidence can be achieved? The answer is one needs to adopt faster means of testing. Many Testing techniques are available which in help achieving this objective. The major drawback of some of the proved techniques is these techniques are dependent on the kind of the system that is being tested. So what are the other ways available to save time for testing? The answer is in the question itself. One can save some time in optimizing the way of testing itself. If we carefully look into each testing activity, we realize that we can improve these to a great extend to provide maximum output. In this paper, it is intended to showcase one such approach.

Keywords : Test Management, Defect tracking, ,SDLC MODEL, Test planning.

GJCST-C Classification: B.8

Approach to Quality Testing

Drakshaveni G

Abstract - Time is the most important resource for any activity. Software development is no exception. With an increase in the competition in the market, it is very essential for the companies to release their products into the market at the earliest with good quality to earn profit. In order to develop the products early, companies implement various techniques like Rapid Application development and implement Agile methodologies like Scrum & Xtreme Programming to obtain tangible and useful features of the software at the earliest. It is just not sufficient to develop the product, it is more important to develop a quality product. But how does one know if the product is of good quality or not? This question is best answered by the Quality Assurance team which Tests the product end to end against requirements and standards. They conduct various kinds of tests and check the behavior of the system/ product in various conditions. If it passes all kinds of tests (which is dependent on the kind of the product or the system), then the QA team assures that the product is fit for use. Hence it is very important for the QA time to be given enough time to test the product/system before it is released into market. But it is difficult to judge as to how much time is required to test a product / system completely. An ideal answer would be "Years". Time is a major constraint in any software activity. Many software projects are time bound. Then how does one ensure that within a given time, the product/system is tested to the level where a confidence can be achieved? The answer is one needs to adopt faster means of testing. Many Testing techniques are available which in help achieving this objective. The major drawback of some of the proved techniques is these techniques are dependent on the kind of the system that is being tested. So what are the other ways available to save time for testing? The answer is in the question itself. One can save some time in optimizing the way of testing itself. If we carefully look into each testing activity, we realize that we can improve these to a great extend to provide maximum output. In this paper, it is intended to showcase one such approach.

Keywords : Test Management, Defect tracking, ,SDLC MODEL, Test planning.

I. Introduction

Testing is an important part of SDLC[1]. It is in this phase that the quality of the product is assured. The product is tested in detail against the requirements and standards. This is not as simple as it sounds. There are many phases in a Test life cycle like Test planning, Design, execution etc. Fig 1.0 depicts a typical Testing Life cycle implemented in many projects.

Author : Asst Professor, Dept. of MCA, BMSIT, Bangalore.

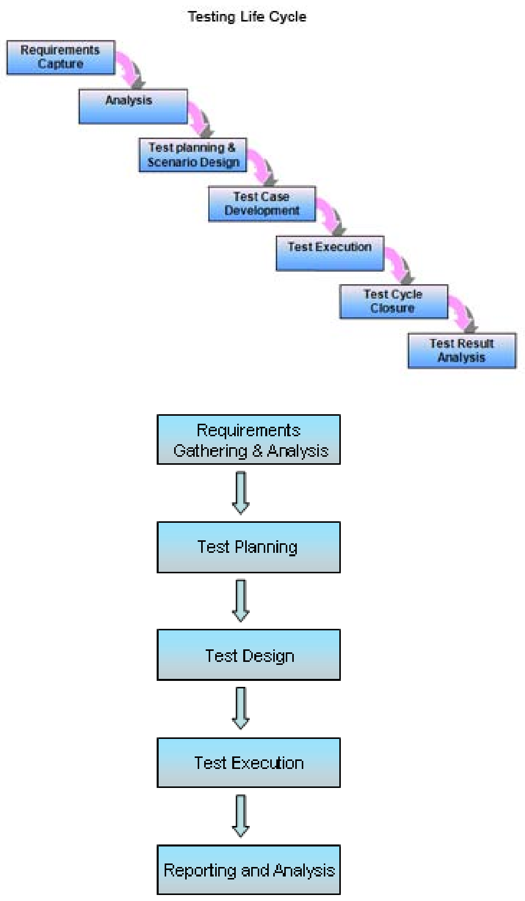

Fig. 1.0 :

When the Requirements are provided, the requirements are captured in a repository using a suitable convention which is easily understood by the stakeholders. Then it is analyzed for implementation. In this phase static testing techniques are implemented. Once the Analysis is complete, Planning comes into picture. This is a complex phases. It is in this phase that the testing approach is designed based on various constraints defined by the project. The success of Testing is largely dependent on the Test Planning & Design phase. Once the planning & Design phase is crossed, then comes the execution i.e. execution of test cases against the requirements and standards. After test execution, the results are analyzed and the quality is measured accordingly. If more test cases / Test scenarios (depending on which one is used) fail, then the product is sent back to the development team to fix the issues. After the issues are fixed, the product is tested again. This process continues till the there are very minimum or no issues detected. If the product is tested and no issues are found, then the product is certified as fit for use.

In the above stated model, each phase has its on set of Inputs, processes and outputs. Any testing activity will be incomplete without supporting artifacts. Hence proper documentation is very important. It is not just sufficient that the product is tested for what it has to do. It is equally important to test a product for what it should not do as well. This is the reason that there will be many input conditions that need to be checked for. Hence testing is a time consuming activity. With Time being a constraint, how to complete so many activities in minimum time? What are the ways in which one can execute the testing activities faster? Let's see few of them.

II. Description

There are many ways through which the testing activities can be optimized. By implementation of these, lot of time can be saved which can be used to execute other activities which yield more result. Few of the most common and useful approaches in this regards are:

1. Transition from Manual to Automation Testing.
2. Usage of good Tools.
3. Forming a team of highly skilled persons.

Let's see how the above approaches can help save time and improve the testing quality.

a) Transition from Manual to Automation Testing

Transition from Manual to Automation is beneficial for the features of Automation. Few of them are Reliability, Repeatability, Fast, accuracy and Cost Reduction [2]. The Automation tools use Test Script language to develop scripts that imitate user actions.

The Test Script Language (like VB Script) is easy to learn and implement. These scripts can be designed to test the applications as a human would do. But the difference is they are faster. They can execute a task as many number of times as instructed without any errors.

As an example, if it takes 1 hour to execute 5 test cases, the Automation scripts can do it in 30 min (Depending on the way the script is developed and the application behaviour).Hence those features of the application which are very stable and require repeated testing, can be automated. The Manual testers can use the same time to test other parts of the application where more defects are likely t be found. Hence more testing can be done within available time. Hence the product can be tested better.

b) Usage of good Tools

Usage of tools like Test Management Tools [3], Defect Tracking Tools and Configuration tools, helps in achieving testing objective faster. These tools have templates/ formats which can be used readily. For example, Quality Center (a Test Management Tool) provides a single application for management of requirements, Test cases, execution and defect management. The reports generated from these tools give a single window data of the status of Testing. This can help the project manager decide the quality of the product. Without this, a subjective decision would be required to be taken regarding the quality of the product, which is a risk by itself. If data is required then it would take a considerable time to collect the same and put it in a format which is easily understood by the stakeholders. Many important Testing artifacts can be stored in such tools which provide the inputs at the execution time. In absence of such tools, the testers would have to refer documents from various sources and when the execution is at its peak, this takes lot of time.

The defect tracking tools help save some time by intimating the right persons at right time irrespective of their location. Example, if a defect is logged, the developer is intimated instantaneously through a mail or an alert, with all the required details as provided by the tester. In such scenarios, most of the times, the developer need not get in a discussion with the tester to know what the defect is all about. Also he can communicate with other teams instantaneously with the all required information. This saves lot of time.

c) Forming a Team of Highly Skilled Persons

This is a time tested approach [4]. Having team with people who know the application/product/system better are always beneficial. They can help the team members by guiding them in the following areas:

a. Implementation of special testing techniques like Orthogonal Array Technique, branch Testing etc
b. Usage of Tools in an optimum way.
c. Usage of right inputs to test the application
d. Understanding the domain and the purpose of each testing phase.
e. Understanding the requirements, Test processes, Test Reports etc.

In absence of a proper guide, one needs to spend more time in understanding the things. With time constraint, if a tester can't understand the full functionality, the functionality can't be tested properly. This is a risk. And if the defect is found at later stage, it would cost more to the project to fix the same. At the same time, there should be new people in the team who can test the product as an new user would do. They might have a new way to seeing a functionality which might help in testing it better and making it better. Testing is a creative activity which requires a good understanding of the application, domain, Testing principles and new ways to thinking. Hence the team should be a combination of people with such mentality so that optimum testing can be done and more defects can be detected at early state. This way rework can be reduced to some extent and hence the time.

III. Challenges

Till now we saw how to save time using Automation, Tools and right team, let us look at some of the challenges in implementing these approaches.

a) Automation

a. Availability of skilled professionals - the newer the technology, tools, methods, and domain, the smaller the pool of skilled professionals
b. Stability of implementation technology - the newer the technology, the lower the stability and the greater the need to balance the technology with other technologies and manual procedures
c. Stability and power of tools - the newer and more powerful the development tool, the smaller the pool of skilled professionals and the more unstable the tool functionality
d. Effectiveness of methods - what modeling, testing, version control, and design methods are going to be used, and how effective, efficient, and proven are they
e. Domain expertise - are skilled professionals available in the various domains, including business and technology
f. Good Automation Tools are costly. It might not be possible for all projects to afford Automation as per the budget defined for the project.
g. Availability of resources trained in usage & implementation of Automation tools. It is not easy to get people who are well versed in usage of these tools. If they are, then they need to be paid more. This again puts constraint on the project budget.
h. The new resources involved in automation, need to be trained in usage of the tools which itself consumes time & money.
i. Automation can be implemented for those applications which are stable. If the application is not stable, then automaton can't be implemented effectively.

b) Tools

a. Good Test Management, Defect Tracking & Configuration Management Tools are costly.
b. Availability of resources trained in usage & implementation of Automation tools. It is not easy to get people who are well versed in usage of these tools. If they are, then they need to be paid more. This again puts constraint on the project budget.
c. Team members need to be trained in usage of the tools which itself consumes time & money.

c) Forming a Highly Skilled Team

a. Difficult to judge the expertise in the required area. Complete knowledge cannot be checked in just a couple of interviews.
b. Availability of experts employment of such people is bit difficult and as they look for high salaries which puts a constraint on the budget of the project.

IV. Proposed Solution

Every problem has a solution. Most of the times, the solution to a problem is very much available but with some search. Having seen the difficulties in implementing the time saving techniques, let us see if we can save some time in the testing process itself. For this, we need to understand what a tester does. Here are some of the steps which most of the testers follow:

1. Understand the requirements and develop the test scenario.
2. Derive useful test cases out of these test scenarios.
3. Execute the test cases and verify & validate the features of the system.
4. If there is any discrepancy observed, confirm it and log a defect.
5. Once the defect is fixed, retest and closed the defect if it is fixed.
6. Capture the results of test case execution and use it further for metrics and analysis.

It is possible to help the testers save some time in step 4 stated above. Normally when a discrepancy is observed, a good tester captures the basic information like how the defect was detected, what was the defect, screen shot of the system (if possible) and associated details. Along with execution of the test case, the tester needs to spend some time recreating the defect to confirm the behavior. Lets take an example.

Suppose a tester is executing a test case for a web based application. That test case has 12 steps and it takes 20 minutes to execute the test case. When the tester finds that step 6 is failing, he logs a defect in the Defect Tracking Tool. He first needs to capture the screenshot of the application to let the developer know as to what is the defect. Then he needs to write the steps to reproduce the defect. It should be elaborate enough for the developer to understand the steps of re-creation the defect. Then he needs to provide some information like Version of the application tested, date & time at which the defect was detected, actual result of the step and expected result etc. On an average depending on the expertise of usage f the defect management tool and the application knowledge, it can take somewhere around 10 to 15 minutes to log the defect with all the required details. This is a considerable amount of time when compared to the test case execution time. This does not stop here. As a good practice, all steps of the test case should be executed to be sure that there are no defect goes detected. If there is another defect detected in step 9 of the same test case, again some considerable time and effort should be spent for defect logging. In such a process, the time spent on defect logging is more than the test case execution time itself.

Here is a method which can help reduce the defect logging time so that the tester can use the same time in execution of other important test cases. This is a solution that should be incorporated in the tool form which the test case is being executed.

1. Make sure that a test case has only on e scenario.
2. If a step in the test case fails, the tool should capture the following details and store it in a repository

a. Screenshot of the application.
b. Step description and expected result of all the previous steps in the 'steps to reproduce the defect section' along with expected result of the step that failed so that the developer can fix the code accordingly.
c. Time stamp information. (I.e. Date and time on which the defect was observed)
d. Version of the application that is being tested (Build number). This information can b entered in some file from where the tool can pick this up.
e. The test data that was used while execution of the test case. This information can be captured by making sure that each test case has a unique test data available at the time of execution. This information can be stored in a separate file with Test Data & Test case mapping so that the tool picks up this information when required.
f. Severity of the issue. This should be defined during the test case design for each step. The test case designer should know the impact of failure of the step that is being executed, on the application.

The above points are just few of the information that can be captured. Other mandatory information can also be captured.

3. Once all the above information is available, the tool should format the information in a way which is easily understood by all stakeholders and should present it to the tester who wishes to log the defect (can be in form of a pop-up window).
4. Depending on the time availability, the tester can review the captured details before submitting the defect or can add extra information, if required.

This way the process of defect logging which takes 10 to 15 minutes, can be reduced to just couple of minutes. If a tester executes 15 such test cases, and on an average he detects 4 defects, he can easily save 30 minutes (approximately) which can be used for adhoc testing or execution of some other important test case. If there are 5 testers, who perform a similar job, in the above scenario, 2.5 hours can be saved.

V. Advantages of the Above Method

1. Time saving: The process of defect logging becomes faster. Instead of taking 10 minutes, the defect can be logged in a couple of minutes.
2. Since the information that needs to be captured becomes standardized, chances of missing out details are reduced to a great extent.
3. Since the information is captured immediately the timestamp information is more accurate which helps the developers to check the logs as to what exactly happened at that instance of time.
4. This method necessitates that the test case be understandable, logical and complete. This helps the new resources of the team to execute the test case easily (which otherwise would have required more time to execute the same). So considerable time is saved and the tester would feel more comfortable.
5. Along with this, it also necessitates defining the impact f failure of each step, which prevents subjective decision on the assigning the severity to the defect which is to be logged for the failure of that step.
6. Depending on the number of High or Medium or Low severity steps, the severity of the test case can be defined. Depending on the number of High or Medium or Low test cases, the Severity of the Test Set can be determined. This is helpful when the testers do not get sufficient time to execute the complete regression cycle and need to prioritize the test case execution based on the severity of the Test Set / Test Case

The advantages are not limited to the above. The availability of extra time itself defined the usefulness of this method.

VI. Challenges in Implanting This Method

Although there are many advantages of this method, there are a couple of drawbacks too.

1. This method calls for use of a tool which can capture all the required information and translate it into useful information as contained in a defect log. This involves time, effort and cost. The project might not have so much time or money to spend on development of such a specialized tool.
2. This method requires defining more details like severity of each step of a test case, capturing unique test data for each test case etc which requires more time and effort.

VII. Where Can This Method Be Useful?

The approach proposed in this paper can be added in any Test Management Tool which has Test case execution and Defect Management features. This approach will be helpful in testing projects where Defect Finding Rate is high and more test cases need to be executed in a shorter time.

VIII. Conclusion

Testers should be given sufficient time to test the system/ product/application in order to get a good quality product. When there is a time crunch, projects should optimize the way testing is carried out. The method stated in this paper can help the Testing project irrespective of the SDLC model followed, to save reasonable time which can be used for performing other important testing activities.

Acknowledgement

We wish to thank all the authors for the providing the informational support

References Références Referencias

1. Testing Common Body of Knowledge.
2. Software Engineering-Ian Somerville.
3. A very comprehensive book on the testing techniques.
4. Johnson, M., Ho, C-w, Maximilien, M., and Williams, L., Incorporating Performance Testing in Test-Driven Development, IEEE Software.

This page is intentionally left blank