10 Best practices Quality Engineers should follow for executing PDLC in a successful way

6th Annual International Software Testing Conference in India 2006

February 17- February 18, 2006
New Delhi, India

Authors
Neeraj Kumar Gupta (neeraj@adobe.com)
&
Abhishek Talwar(abhishek@adobe.com)

Adobe Systems India Pvt. Ltd.
Noida

Abstract

“10 Best practices Quality Engineers should follow for executing PDLC in a successful way”

While developing a product one comes across many situations in which a person needs to take decisions which might present all alternatives which seem wrong or all alternatives which seem correct. In case of such tough decisions, one has to be guided by some best practices which can help him to solve this problem if not present a readymade solution to the problem. Our Paper would try to present some of these best practices that can help us in making decision making at different levels easier.

We would like to present the best practices in two broad categories:
1) 10 must haves for a Quality Engineer
2) Four things which any professional or entrepreneur cannot afford to forget

The best practices of a Quality Engineer ensure that quality of the final product is ensured.

I. Know the technology
II. Learn from past experiences
III. Ensure sufficient coverage of feature workflow
IV. Make a matrix for hardware to be supported by application (and prioritize the hardware)
V. Assign two QEs (primary and secondary)
VI. Plan real life resource intensive tasks in advance and procure resources accordingly
VII. Test smartly
VIII. Track when functionality is broken (Sanity testing on every build)
IX. Feature sweep, hardware sweep, workflow sweep for every feature
X. Proper documentation

Other than the qualities mentioned above specifically for Quality Engineers there are some qualities that not just Quality Engineers but every person working on any project must have:

i. Be passionate about the stuff
ii. Be willing to learn
iii. Understand and make your customers happy – “Customer is the king”
iv. Study the competitors

Introduction

Testing is a major activity in any development cycles. A good tester makes sure that application being tested meets requirement of customers. In an ideal situation there should not be any bug in the application but this is only the case in ideal condition. In any medium to large software project there are some bugs, which might have been incorporated during requirement gathering, designing, and coding a unit and in integration.

A good tester makes sure that proper testing planning and control will identify as many bugs as possible. This paper would try to answer questions regarding the best practices not just from the QE perspective but also the best practices that every person on the project should keep in mind to help smooth sailing of the project.

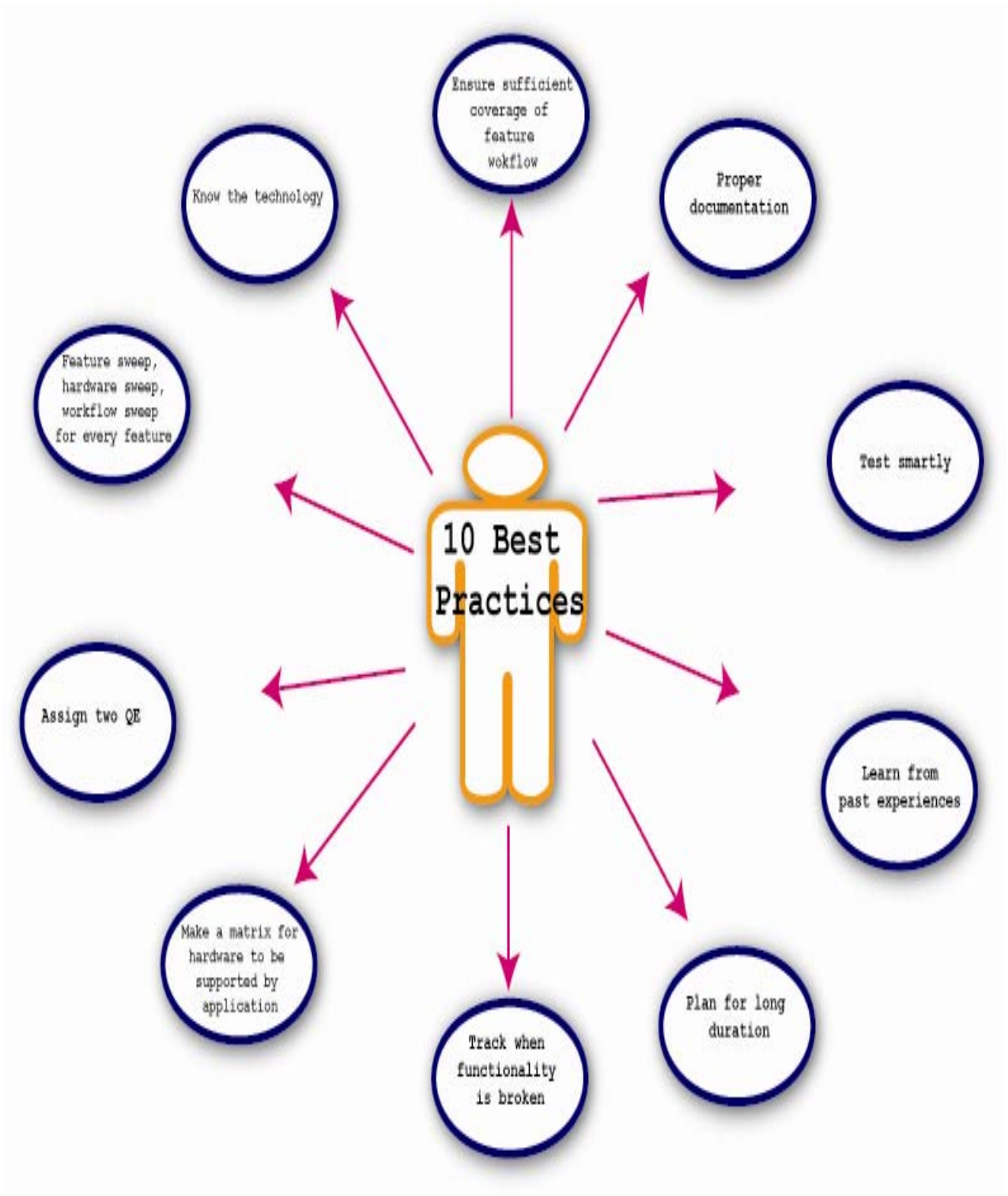

Figure 1: Pictorial diagram for 10 best practices

1) Know the technology

In the area of software development, technology is changing rapidly. What is new trend today may become obsolete tomorrow. In this fast changing world a tester should keep abreast with the latest in the technology world. Tester should know what technology is being used for developing application. He should know supporting OS and technology associated with OS. This will make sure that tester will find bugs as early as possible, thus reducing the price for the fixing of bug.
Also, studying the technology helps study the competitor better and thus helping your product to have a good standing in the market. Studying the technology would also help the Quality Engineer to understand the workflow and the product itself better.
In the present scenario when the technology gets outdated faster than it is introduced the testers need to be on their feet to make sure that the technology if relevant to the product can be fully utilized.

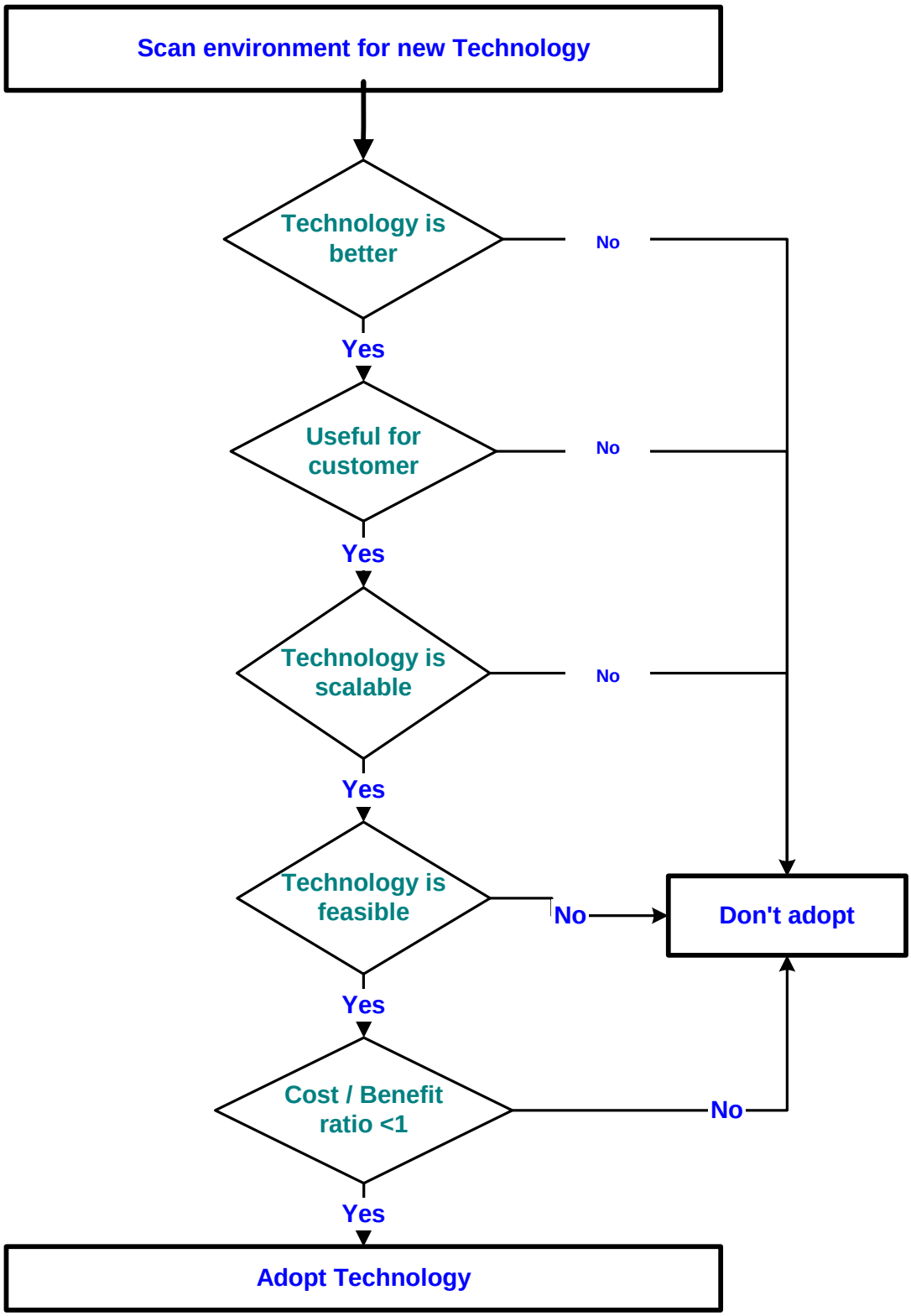

Figure 2: Adaptation of new technology

2) Learn from past experiences

Before writing test plan and test cases, QE should try to scan data from past projects, which are similar in nature. This may give them some better test cases, issues that were identified in the end of the cycle, some area that could be automated.

If the application is extension of earlier project, high priority and severity bugs should be regressed for current version of application.

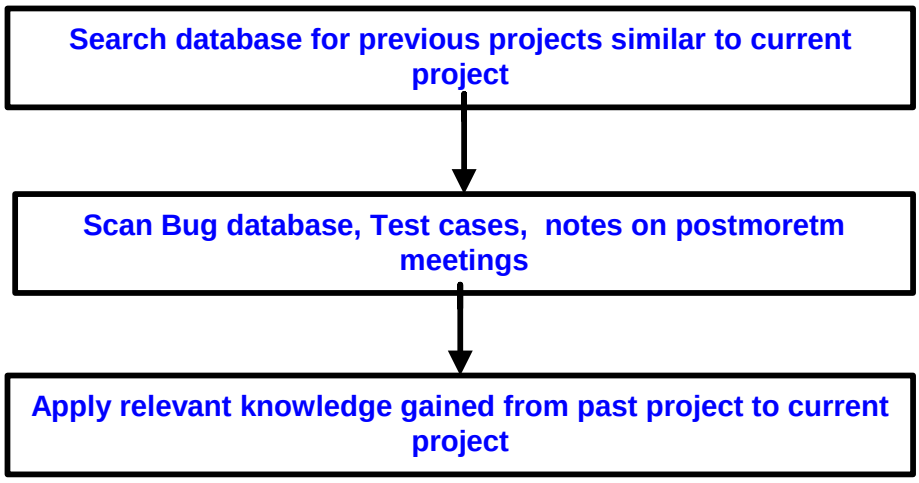

Figure 3 Learning from past experience

3) Ensure sufficient coverage of feature workflow

The key to success of a testing effort is that when a customer users the product he/she should be able to seamlessly use it. This kind of seamless experience can only be ensured if the common user workflows have been tested thoroughly. The challenge here is that different users use different features and in fact might even use the same feature in a different way. Here is where the role of sufficient coverage comes in.
Coverage can broadly be classified into two categories:

a) Coverage of different functionalities present in the software

The software performs a series of tasks. However the usage of these tasks may vary depending on the requirement of the customer. e.g. the most common image editing software Adobe Photoshop is used by different sections of the industries. Students use it to make their projects and assignments, digital photographers use it for touching and editing in their images, Animators use it to generate series of stills to be converted to the animation. For each of these people one single software, namely, Adobe Photoshop is the solution. In such a case, one cannot just ignore one functionality and focus on the other. i.e. for Adobe Photoshop would be used by a school/university student to make illustrations for the assignment and in that case he would start from scratch and work on object like squares, rectangles etc. While the digital photographer would use the same software to give fine touch to the existing image and work on it to get his final image that he can use for an exhibition. While an animator would have to

scan the image of a cartoon character and start to work on giving him actions. Once the possible workflows have been identified, one needs to prioritize the workflows depending upon the number of users who would use the particular feature.

b) Coverage of the different routes while going through the workflow of the software

Common functionalities can be accomplished in many different ways in most software. E.g. in order to copy a text in Microsoft Word application, one can use the keyboard shortcut Ctrl+C, or right click on the selected text and select copy. One can also use the edit menu to do the same. Each of these three routes is important because user can copy text using any of these. As a tester one needs to find all possible routes of accomplishing a task. After the routes have been identified, Quality Engineer needs to see how much probable are these in a real world scenario. Also study of usage of this feature in comparison to other features needs to be done. As for our example, we need to know how much common it is for the user to use the 'copy' feature in comparison to other features like 'find', 'replace', 'cut' etc.

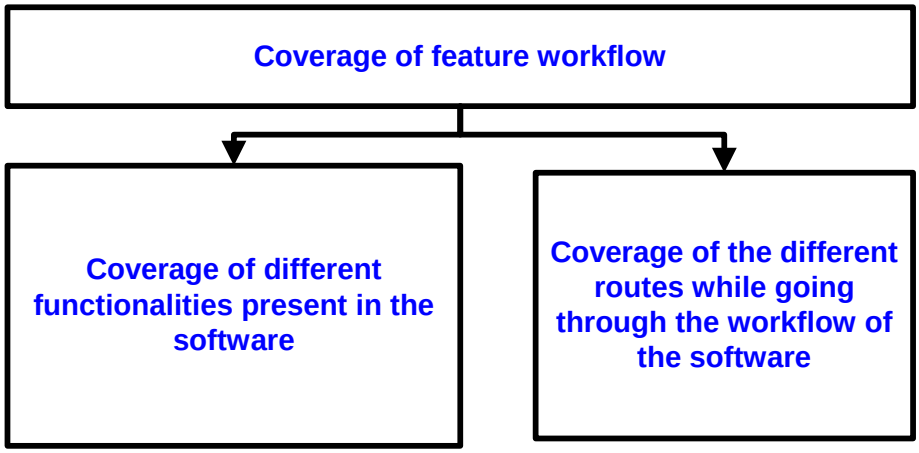

Figure 4 Coverage of feature workflow

4) Make a matrix for hardware to be supported by application (and prioritize the hardware)

Many applications support different configuration of hardware. A proper matrix ensures that application is working on important configuration of hardware.

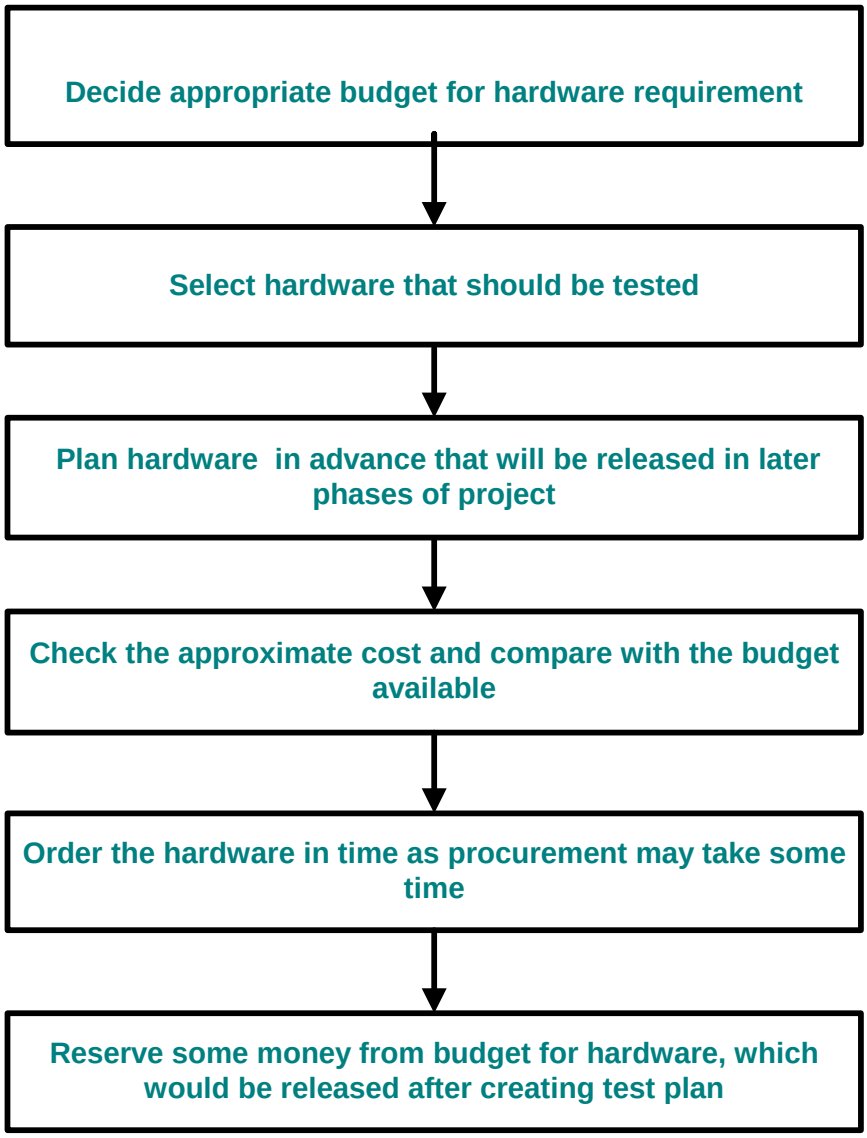

Figure 5 Selection procedure for hardware

5) Buddy QE system (primary and secondary QE for each feature)

An application has multiple features. There should be two QE assigned to a feature. The QE can be assigned as primary QE and secondary QE. Primary QE is responsible for creating test plan, test cases, setting test environment, executing testing, reporting bug and regression of bugs. Secondary QE can work as a back up. He can be involved for reviewing test plan and script, helping primary QE in setting up testing environment, executing some part of test cases at different interval and in regression of bugs. This will help in getting control of bug myopia (blindness to a bug if working for a long time in testing a feature).

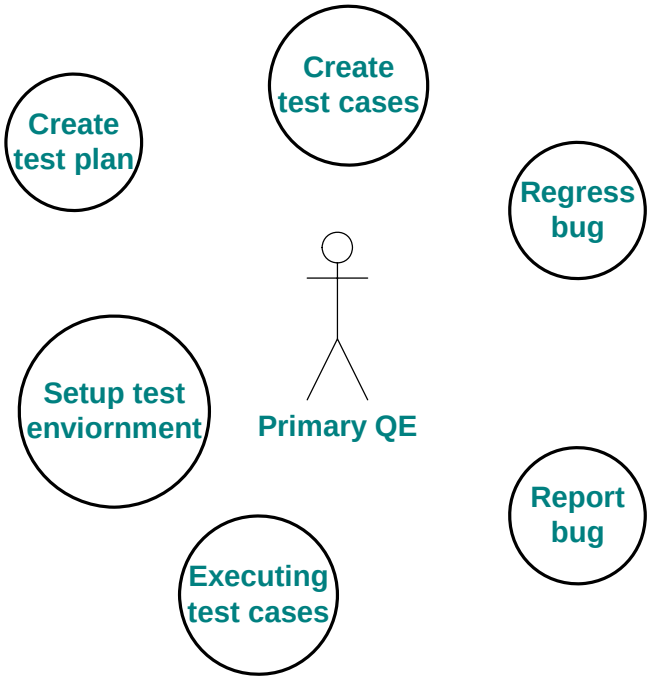

Figure 6 Role of Primary QE

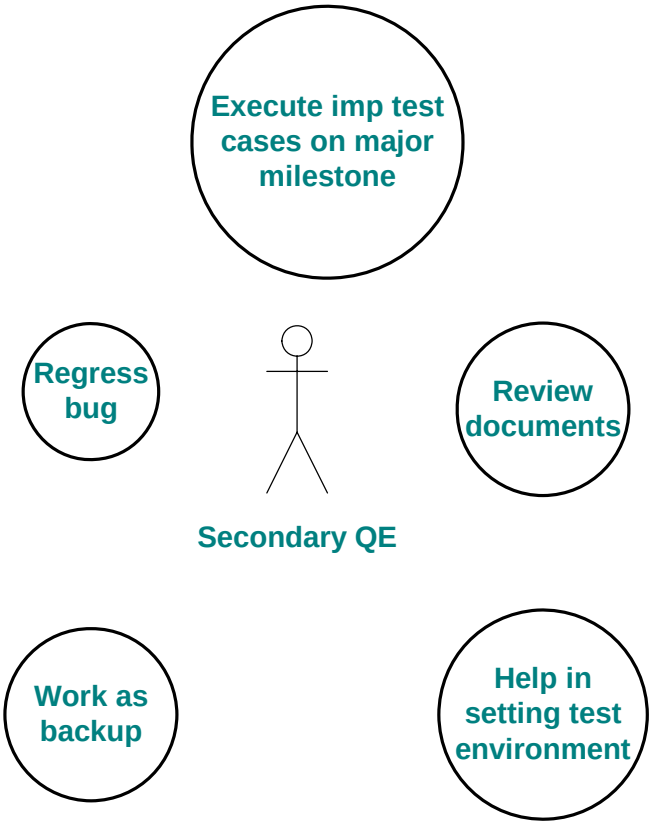

Figure 7 Role of Secondary QE

6) Plan to use a true representative set of data from users during testing

Some applications may require processing large number of data for final output. During functional testing of such applications, for quick results, a tester keeps on doing testing with small set of data, which may take very less time. But user of that application will work with real data, which may take significant time. A QE should process real time data that requires application long time to process the data. Testing such cases in the end is not good idea. A failure for long duration run is difficult to isolate. Isolating of such bugs is time consuming activities. In the end of cycle such bugs will create panic and may delay major milestone.

Proper planning ensures smooth testing for long duration run. Long duration tests can be executed during night and weekend. In case there are common systems, with test environment, procuring will be easy if planning is done in advance.

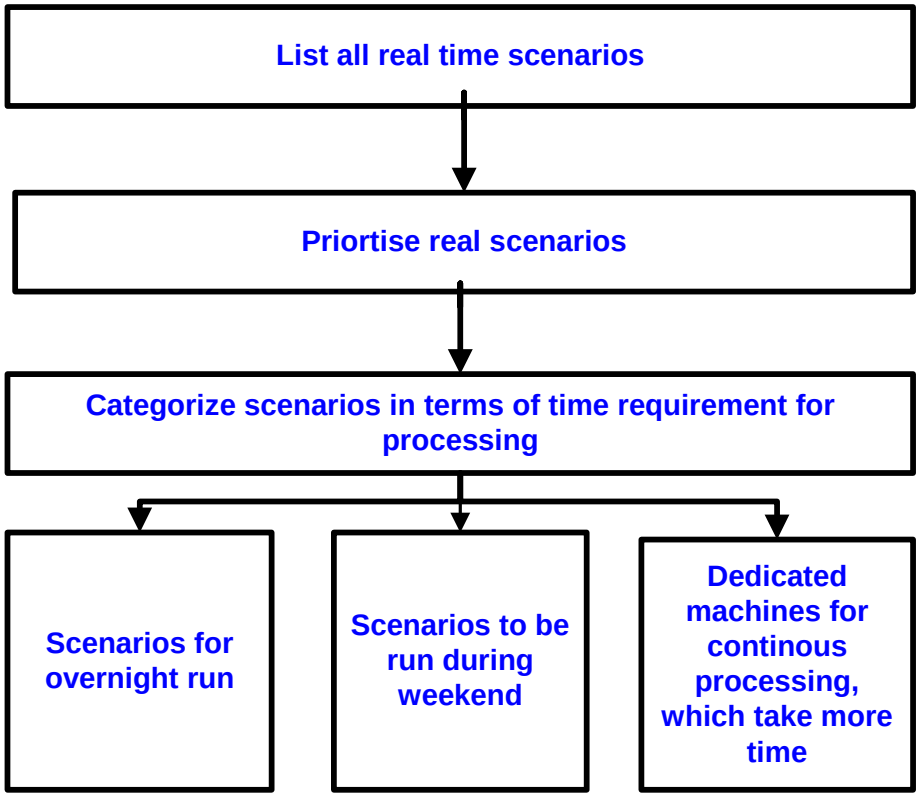

Figure 8 Categorization of real time data

7) Test smartly

As products become more and more complex, testing them becomes a bigger challenge. By no means can one test every possible value that a control can take. Some of the smart testing approaches that one can adopt are:

a) Automation

Automation helps a lot in covering a lot of regression features thereby reducing the overhead of the test team to test out the old features. Automation is very useful for

resource intensive tasks as it ensures that no machine hours are wasted and limitation of the hardware is avoided.

b) 80-20 rule

This law states that eighty percent of the functionality should be covered in twenty percent of the test cases. This method is a very handy resource of saving time and helps to cover the most important scenarios as fast as possible.

c) Test buddy

Discussed in point number 9

d) Focus and encouragement on catching only crucial bugs early in the cycle

The cost of fixing a bug early in the cycle is very less in comparison to the bug logged late in the cycle. In fact it is said that cost of fixing the bug at the end of cycle is 100 times more than fixing a bug at the start of the cycle.

e) Study relationship between components

If relationship between the related components is known the overall overhead cost on the project reduces. Similar resources can be combined and a better utilization of the resources can take place.

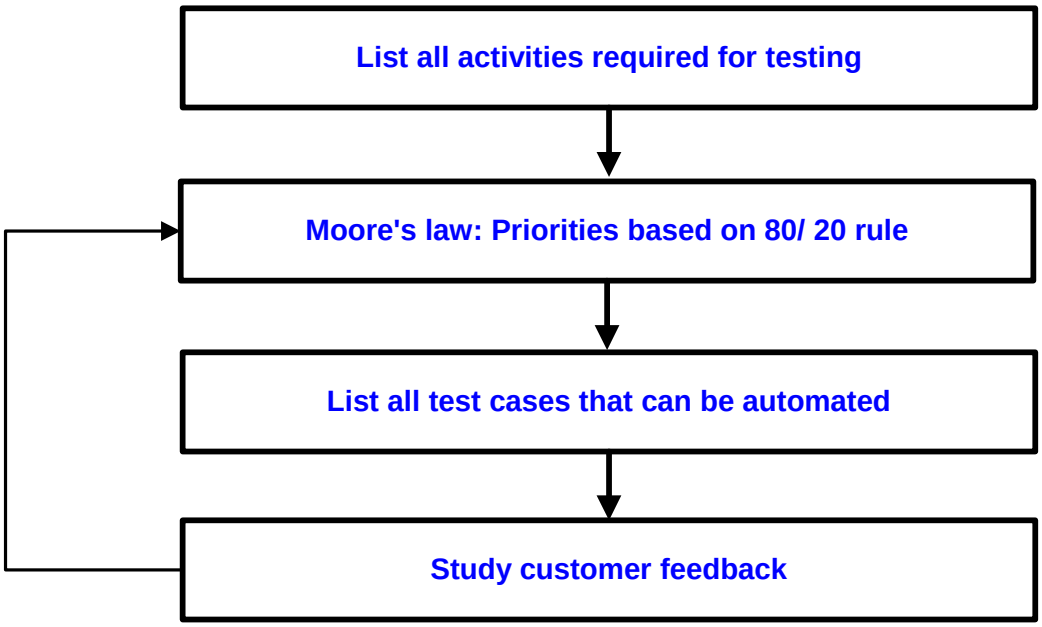

Figure 9 Prioritizing test cases for testing smartly

8) Track when functionality is broken (Sanity testing on every build)

After complete testing QE may think that since there is no change in that particular code, there is no problem. In large application, change in the code of one area may break other functionality also. QE should make a test script with high priority test cases. QE should ensure that with every new build release to QE, this test script should be executed to ensure that there is no problem. At the end of the cycle the frequency of the build increases significantly, if it is not possible to execute all the tests for every build the tester can execute the sanity test on alternate builds & update the test case document with the build tested. This will help to track down when the functionality was broken.

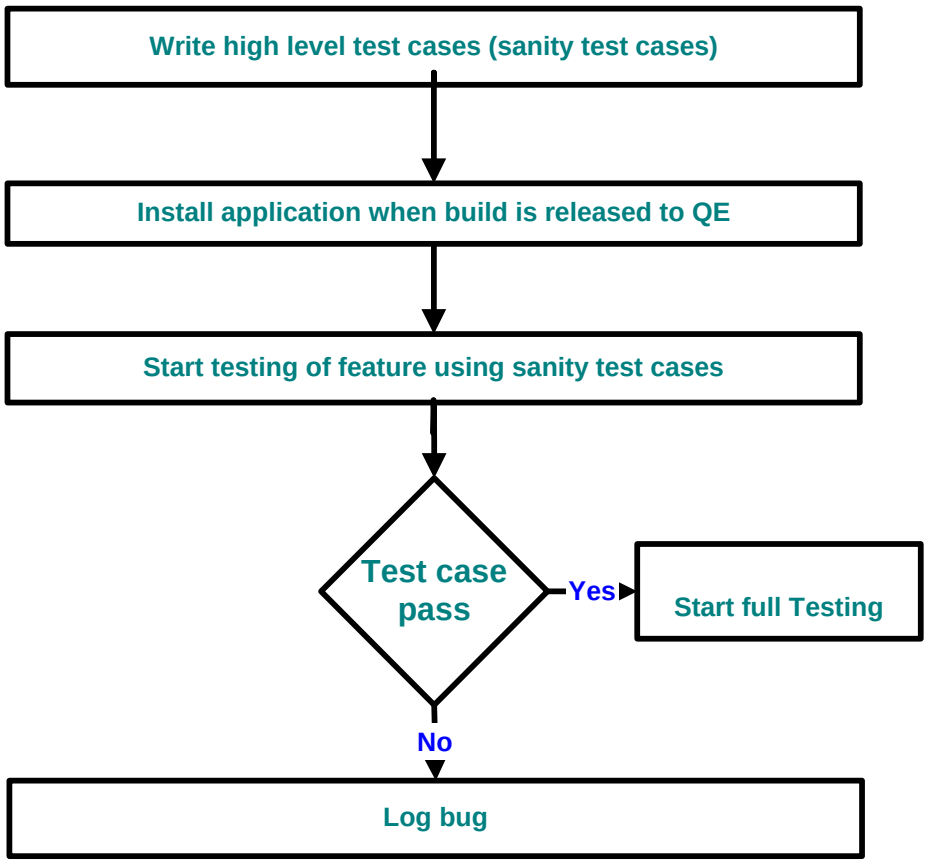

Figure 10 Sanity testing for feature

9) Feature sweep, compatibility sweep, workflow sweep for every feature

a) Feature Sweep

Feature sweep covers basic test cases of functionality. For every major milestone, these test cases should be executed to ensure that functionality is not broken. It is also being observed that if a QE other than owner of the feature will execute the test cases, it would yield more result.

b) Compatibility Sweep

There are many features that require test cases for multiple hardware platforms, Operating System among others. For a application that is supported on multiple OS (e.g. Windows, Mac OS), different hardware configuration (AMD, Intel, Dual processor, hyper threaded machine, different RAM etc.) and hardware (e.g. different printer for printing application) such sweeps are very important.

c) Workflow Sweep

Owner of a feature in an application execute test cases for testing functionality of feature. By doing this functional area of the application gets covered by testing all features by their respective QE owner. But a customer perspective may be missed. It may happen that all functional areas are working but common workflow has hindrance. There may be usability issue, performance issue; some inconsistent crasher may come

due to memory leak in some part of code. Major workflow of a product should be identified & assigned to the QE team members. A QE should take this assignment very seriously & include it in his/her weekly reports. If time permits he should take a look at other workflows

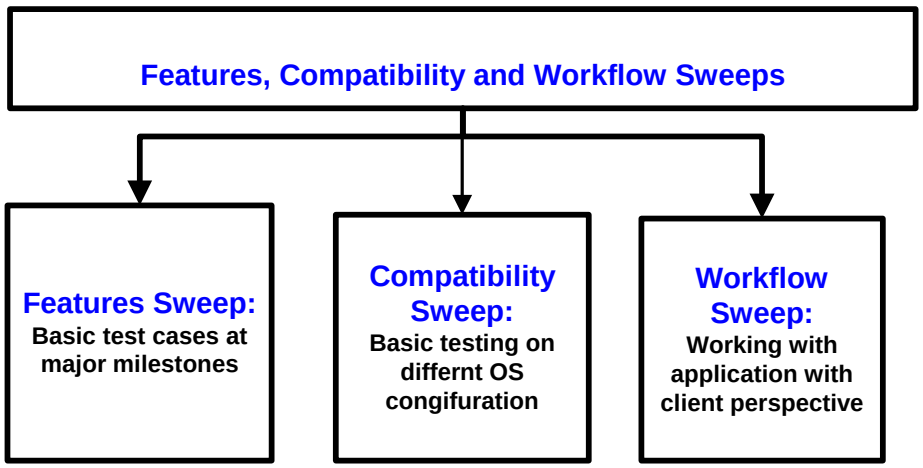

Figure 11 Features, compatibility and workflow sweep

10) Proper documentation

Documentation is a very wide term and includes any kind of written communication that helps the project in sailing through. Documentation can be classified into four broad categories:

1) *From the developers point of view*

The documents are

- Feature spec
- Design document
- Commenting

2) *From the testing point of view*

It includes:

- Test plan
- Test script
- Execution matrix
- Resource planning document
- QA plan, among others

3) *Documentation meant for the external users.*

This includes:

- User guide
- Help files
- Readme documents

4) Project management and Product management

- Requirement Specs

- Project Schedule

As a good Quality Engineer one should try to do the right amount of documentation ensuring that the documentation helps smooth execution of the project and also helps to catch bugs early in the cycle. Too much of documentation turns out to be an overhead to project and can lead to slippages in schedule or even insufficient testing.
Other things to take care for while documenting are, that it should be scalable to meet even the future demand and also ensure that it doesn't become an overhead.
Documentation is also important because it gives the management an idea on how the project is progressing.
Hence documentation done in the right amount would certainly help in the overall progress of the product.

It is also important that the test documents get updated as your product evolves. It is a good idea to update your test documents at important milestones of the product

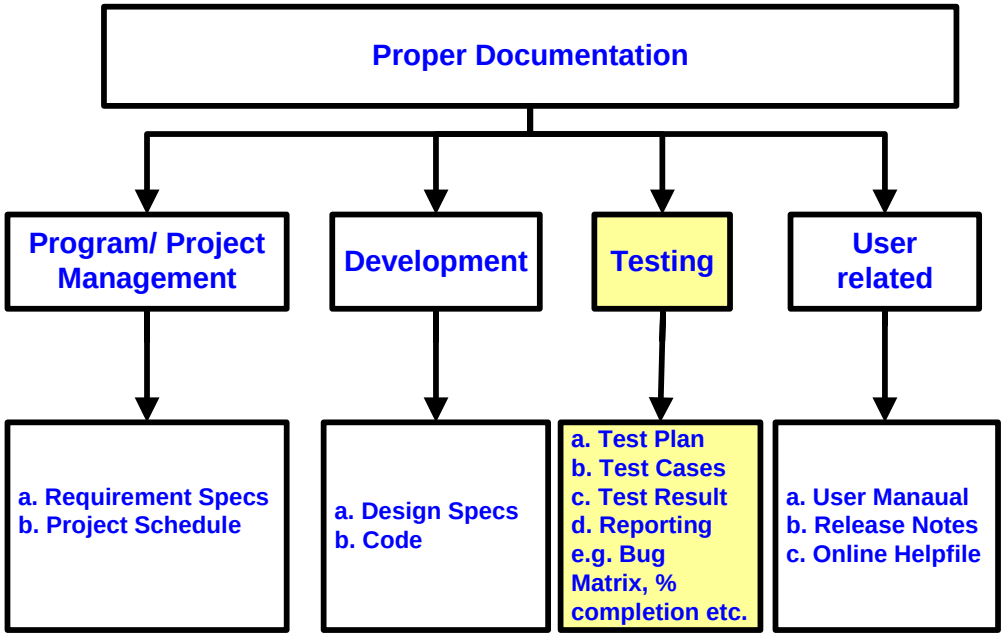

Figure 12 Different documentations created in typical cycle

Four best practices for any person on the project

Other than the qualities mentioned above specifically for Quality Engineers there are some qualities that not just Quality Engineers but every person working on any project must have:
In any project, role of every person is important. There are some best practices that are not for Quality Engineers but for all working professionals and entrepreneurs. A Quality Engineer knowing these is a definite plus on all counts. Following are the four important ones:

1) *Be passionate*
   A person who likes his work can rise to greater heights even with lesser knowledge than a person who knows more but is not passionate about what is he is assigned to do.

2) *Be willing to learn*
   Learning is ongoing process and one who is open to learning would always succeed
3) *Understand and make your customers happy*
   Customer is the king
4) *Study the competitors*
   If a person knows what the competition is then he can work better and make things not going in favor into things working for him.

Neeraj
Neeraj K. Gupta is Quality Engineer at Adobe. He has over all experience of 6 years in the field of software testing. He has worked on various domains like Video, GPU card related testing, Insurance, Risk Management, Earthquake modeling, Remote Sensing, GIS, and Transportation Engineering. He has published three papers in QAI International testing conference 2004 and 2005 on topic like Testing Risk Management Application, API testing and Device Dependent testing, which were selected for presentation in the conferences. He has also published papers on ArcObject (ESRI product) in Risk Management field in ESRI User Conference 2002.
He was also part of process definition group for CMM level 3 and was playing role of SQA (Software Quality Analyst) in his last company.

Neeraj holds Bachelor's degree in Civil Engineering and Master of Engineering degree in Civil from IIT Roorkee (formerly, University of Roorkee). Also, he holds one year certification in Russian Language from IIT Roorkee. He is also ISTQB Advanced level certified tester and Adobe Certified Expert in Premiere Pro

Email: neeraj@adobe.com

Abhishek
Abhishek Talwar is Software Quality Engineer at Adobe. He has experience of 3 years in white box testing and black box testing. His domain of work include: Digital Video, Digital imaging and Telecommunications. He has also worked on analyzing memory leaks and other kinds of errors related to resource usage through Bounds Checker. His areas of expertise include helping different teams with BoundsChecker. He has also worked on Scripting on After Effects. Prior to his 2+ year stint at Adobe he worked on a project of a US telecom giant for integrating their automation solution into a single entity ‘sniff test’. He has submitted 2 papers at the last year’s STC one of which (API Testing: Catching hidden bugs early in cycle) was presented at the main conference in Hyderabad.

Abhishek holds a B Tech degree in Information Technology from University School of Studies, Delhi. He is an ‘Indian Testing Board’ certified foundation level tester.
Email: abhishek@adobe.com