Practical Quality Management for Project Managers

David Lutzker, CSTE, Consultant, Innovative Consulting Inc.

Introduction

Project Managers need to practice Quality Management, sometimes with a Quality Manager, and other times on our own. In order to adapt the processes and procedures that ensure a quality product, we need to understand the underlying principles. With some background in Quality Assurance, Quality Control and Quality Planning we will be equipped to deliver a quality product to our customers.

In this paper I will present and review the definitions and importance of these basic concepts: Quality, Quality Assurance, Quality Control and Quality Management in Projects. I will provide details of Quality Control techniques and best practices, including Requirements Testing, Test Strategies, Test Plans, Test Cases, Reviews and Inspections and the difference between Quality Control and Quality Assurance.

Getting Started

In order to give the Project Manager the tools to ensure a high-quality product, we need to cover some of the basics of Quality and Quality Assurance.

Quality Defined—A product is a Quality product if it is free of defects (Quality Assurance Institute, 2001).

This is a theoretical definition, since people are creating the product. One of the cardinal rules of Quality Assurance is that there is no such thing as a defect-free product. It is, however, valid as a goal. To make it more tangible, we will further qualify by the audience: Customers (or Users) and Suppliers (or Producers):

Quality for the Producers—Meets Requirements or Specifications

Quality for the Users—“Fit for Use.” Fit for Use is a term used in Quality circles to mean how well the product meets the needs of the users. Theoretically, if the requirements were drawn up in a perfect world, the degree to which the producers met the requirements would be the same as the degree that the product is fit for use. In this practical discussion of quality, we understand that there is always a gap between these concepts, to one degree or another. One of the aims of Quality Controls is to reduce that gap to the minimum.

Quality Assurance (QA)—The set of support activities needed to provide adequate confidence that processes are established and continuously improved to produce products that meet specifications and are fit for use (Quality Assurance Institute, 2001).

Note that these are “support activities, they are not the primary activities of the project. QA is more process-focused, and less product-focused. It exists for the sake of management, not for the sake of the project team. Management is concerned that the processes are being employed that will be repeatable on subsequent projects. The support activities are used to increase their confidence that this is so. Also note the use of the dual definition for the quality product, “meets specifications and fit for use.” QA has a slightly different focus in the Software Engineering Institute’s (SEI) Capability Maturity Model (CMM).

Quality Assurance (SEI)—CMM Level 2—The purpose of Software QA is to provide management with appropriate visibility into the process being used by the software project and of the products being built.

One important difference between the two definitions is the SEI’s focus on visibility. Whatever activities are being designed by management, their purpose is to get inside the processes that are being employed by the project team. The continuous improvement aspect is included in CMM Level 3.

Testing—Examination by manual or automated means of a system by executing the functions of the system to verify that it satisfies specified requirements or to verify differences between expected and actual results.

It is important to note that the system must execute its functions in order to be tested. Examining the inner workings, documentation, program code or design are all valid Quality Control techniques, but they are not a test of the system. The actual results of the test must be compared to the expected results, based on the requirements of specifications. Testing is executed with the help of a Test Plan, which spells out in each Test Case what is being tested, and the expected results of the test.

Inspection—Assessment of a work product conducted by qualified independent reviewer(s) to detect defects, violations of standards, and other problems. The primary goal is to identify defects, not correct them.

Verification and Validation—These two concepts usually appear together, and are often confused with one another.

Verification—A process used to determine if the product or phase meets the requirements and specifications set forth at the
inception of that phase. It should be used at the conclusion of each phase in the project. It answers the question, “Are we building the system right?” It is primarily a producer-centric process.

Validation—A process used to determine if the product or output meets the needs of the users. Does it conform to user requirements? It should be used at the end of each phase, and at the end of the project. It answers the questions, “Are we building the right system?” It is primarily a user-centric process.

The combination of these two concepts gives a powerful technique for examining the results of a phase or project.

Quality Control (QC)—The process by which product quality is compared with applicable standards, and the action taken when nonconformance is detected. Its focus is defect detection and removal. QC activities include testing, review and inspections and verification and validation. It is primarily concerned with the quality of the project. Contrast this with Quality Assurance, which is more concerned with the processes employed by the project team.

Audit—The audit is an independent examination of the processes and products and their compliance with plans, policies and procedures. Audits serve as the eyes and ears of Management, providing visibility into the project life cycle.

Ensuring Quality Requirements

Having excellent requirements does not happen by accident. It is widely accepted in the industry that requirements defects are the most common, and are the most expensive to fix if not found during the requirements phase. Many defects and flaws that are found during the execution of the project are in fact traceable to ambiguous, incomplete, vague or conflicting requirements. Good Quality Control practices include testing requirements thoroughly during the gathering and documentation process.

Requirements Gathering Techniques

The requirements phase can sometimes be the phase with the most intense contact with the customers and end-users. It is an opportunity to lay the groundwork for your relations for the rest of the project. You are not simply setting their expectations, you also have the chance to build a rapport, and create a network of contacts: official and unofficial.

Requirements Gathering

Plan on using more than one technique to gather requirements. Anytime you look at a problem from two different angles, or the business needs for a system from two different users, you will discover more about the object of interest. The techniques you are planning to use should be documented in the Quality Plan, or the Project Plan if there is no documented Quality Plan. Here are a few ideas:

1. If you are going to interview a group of users, but one person can’t fit it into their schedule, find a few hours to spend with that person one-on-one and shadow them in their day-to-day work.
2. Request to participate in relevant internal departmental meetings and management meetings.
3. Get yourself added to the same email and paper distribution lists in which your users are members.
4. Ask for access to the same online reports and reporting tools.
5. Use the company website and intranet site to learn the jargon that is used in the company.
6. In extreme cases when you cannot schedule a meeting, or even a conference call, circulate a questionnaire by email, and follow-up with phone calls or individual meetings.

Requirements Documentation

I have not come across a situation, or a project so small and insignificant that it was not worthwhile to document requirements. There are many reasons why it is a good practice, but at the very least, it is the most important tool to ensure that you have accurately understood the requirements of the project. The customer must review what you have understood, and what you think you are being asked to build. It must be written in language that they can understand, and in a structure that makes sense. I will mention two methods:

1. Hierarchical—The classic format for a textual representation of requirements. There is nothing wrong with this approach, if it is done properly.
2. Use Cases—Use Cases are borrowed from the world of Object Oriented Analysis and Design. They are simply scenarios of functionality. Each Use Case represents a kind of task that a user will perform with the system being built. I will not go into the details of Use Cases, but I want to point out that you can use this powerful form of describing what a system will do without using other aspects of the Object Oriented approach. You don’t need to use the modeling languages or tools that are typically used by organizations that have adopted the Object Oriented approach in order to make full use of this technique for documenting requirements. I highly recommend it.

Requirements Validation and Verification

Before actually handing the requirements documentation to the customer, you will want to examine them internally for flaws. There is a long list of attributes that excellent requirements possess, including:

1. Concise, clear, non-ambiguous—As written, a requirement needs to be understood in the same way by everyone involved in the project. This is the time to control problem typical of organizations with silos. A term can mean different things to different departments. A concept can be called more than one name by one or more departments. I’ve never written a requirements document that didn’t have a section for definitions of terms and acronyms.
2. Discrete, atomic—Sometimes a requirement needs to be broken down into subrequirements, in order to be accurately executed. Ask yourself, “Can this requirement be broken down?” Break it down until it cannot be broken down any further, just like an atom.
3. Scope—The question of scope should be asked in both directions: Is the requirement within the scope of the project (a test for relevance)? Is the project fully within the scope of the requirement (a test of importance)?
4. Quantified, testable—In order to know if we have met the requirements at the end of the project, we need to be able to test the results of the system, and compare those results to what was requested. If the requirement was, “The system shall complete one cycle in a reasonable amount of time,” your success may depend on whether the project sponsor had a good night’s sleep. Quantifying requirements can sometimes be a painstaking process, but persist,
and be creative. A “user-friendly system” is one which 7 out of 10 new users rate it as “easy” or “very easy” after a one-day training class. If the customer doesn’t want to test it, then perhaps it shouldn’t be included as a requirement.

5. Traceable—Requirements Tracing is an important technique, since we can use it to control scope creep. We should be able to trace each activity in the project to a requirement. Every design decision should be answering a need expressed in one or more requirements. Every module or program written should justify its existence because it is needed to meet a requirement. We also trace requirements “backward”: every requirement should be traceable back to an objective in the project charter or initial project plan. If it can’t, it is probably out of scope. An excellent technique for ensuring traceability is a simple matrix, which can also be used to manage requirements throughout the project.

Requirements Management

Requirements Management is necessary partly because we may not do a perfect job of ensuring that every requirement is of high quality. However, even if we did a perfect job, the customer can announce midway through the project that they have changed their mind (as often happens in real life). That is when a Requirements Traceability Matrix can be indispensable. A sample matrix is provided in Exhibit 1.

Quality Plan or Testing Strategy

Create a Quality Plan or Test Strategy early in the project. Understand what level of quality your customer expects before the project begins. Make sure that it is defined clearly and to an appropriate level of detail.

A Quality Plan is a necessity if Quality is to be managed during the project. As with many aspects of the project, managing the customer’s expectations can be critical. Obviously, the need for high-quality work is vastly different for a website than for a life-support system for the space program. This can be a difficult exercise for a customer who has not considered what level of Quality they require. Do not give in to ambiguous requirements for Quality, as they will inevitably reappear later in the project, at a far greater cost. Along with the requirements for Quality denote the strategy for meeting them. You Quality Plan should include a description of the testing environment and all the Quality Control activities needed to support the Quality effort.

Test Plans

If you have done a thorough test of the project requirements, then your test plan is almost written already. In fact, it makes a lot of sense to combine the two: Create your test plan during the Requirements Phase as a method for testing the requirements. There are huge benefits to be derived from this approach.

Components of a Test Plan

The Project Test Plan is a detailed document that explains exactly how the quality requirements will be met. Most test plans will include these elements: functions to be tested/not tested, exclusions, limitations, a description of the test environment, the testing approach and methodology, the types of tests to be executed and their dependencies, requirements tracing, pass/fail criteria, testing tools to be utilized, resources required, testing schedule, deliverables, risks, contingencies, release criteria and all the test cases needed to complete a full system test. Each test case will test one scenario for one requirement. There need to be negative test cases to prove that the requirement will be met under all circumstances.

The ROI of Creating a Test Plan

A comprehensive Test Plan pays for itself easily. Consider the following benefits:

1. Having a Test Plan keeps the developers, analysts, designers and testers honest.
2. By completing the Test Plan early in the project the developers and designers can use it in their own testing.
3. A Test Plan prevents defects and saves time and money.
Test Cases

Test Cases are the heart of the Test Plan. Each Test Case is one scenario of functionality for one requirement. This may seem a relatively simple task, but if it is to be done efficiently, specialized skills and experience are very beneficial. No project can afford to test every possible combination of inputs into a system. Even when testing is automated, the preparation of the test cases would take longer than the project itself. Using the Pareto Principle (the famous “80-20 rule” states that 80% of the defects are found in 20% of the system) we can optimize our testing effort. Where are the defects most likely to be found? Experienced testers and test designers who are familiar with the project will find them.

Making the Most of Your Testing Resources

Testers need to take the approach that their job is to break the system being tested. While this may seem adversarial, that is their job. An experienced tester knows that if they don’t find the defects, the users of the completed system will. If test cases are being written during the design phase or the beginning of the development phase, then the developers can be made aware of exactly what the testers will be testing in their code. If you have a test environment and test data available, your entire project team can make use of the test cases. They will be inclined to test their work products themselves, and find their own bugs, thus creating substantial cost-savings, and a higher quality product.

Design and Implementation

Most of the Quality Control techniques employed during the Design phase are used in the Implementation phase also. There are industry-specific differences, for example, in software development projects there are various types of program code reviews that can be utilized. This paper will discuss techniques that can be used across many industries.

Inspections and Reviews

Inspections should be made throughout the project life cycle. In fact, emphasis should be put on finding the defects as early in the project as possible. A delay in identifying a defect can raise the cost of correcting it considerably. In Exhibit 2 a graph depicts the approximate relative cost of finding and fixing a defect during the different phases of a project. The graph is schematic, but is based on commonly accepted measurements based on various studies.

If inspections are done regularly and correctly there are numerous secondary benefits that will be realized by the organization. Exhibit 3 depicts some of these.

Making Inspections Successful

In order to be successful, inspections need to be planned. It is critical that the right people be present in an inspection meeting: The author, qualified, independent reviewer(s) and a competent facilitator. It is important that the author’s manager not be present, and that the results of inspections are not included in performance reviews by management. It is best practice to supply the Reviewers with the target of their inspection enough time prior to the inspection meeting to allow them to become familiar with it. The most important thing is to maintain the focus on the primary goal: Find and record defects! The reviewers should be concentrating on the quality of the artifact itself, not on the author. The reviewers must keep in mind that responsibility for a defect ends after it is recorded and verified: they should not fix the defects. Authors have the responsibility to correct their own products. Following the meeting the author will make corrections, and submit the product for a reinspection. The process must repeat itself until all defects are either fixed or accounted for.

The Psychological Aspect of Inspections

There is an important psychological aspect of Inspections. If the psychology involved is not fully understood, the result can be a failed inspection, with serious repercussions to the author, the project team and the organization. Management must take responsibility to educate all reviewers, facilitators and authors on the goals and means of Inspections. All the members of the inspection meeting have to keep in mind their common goal of achieving a high-quality output at the end of the project.
Testing

Even after all the Quality Control activities that have been executed during the project, the testing phase of is still necessary.

Types of Testing—Below is a sample of some of the types of tests that can be employed during a project.

Black box—The test of a system or a component against its specifications and/or requirements without knowledge of how the system is constructed. The inner workings of the system are not examined.

Performance—How quickly does the system respond?

Stress—At what point does the system fail?

Documentation—Give the documentation to an actual representative of the target user population.

Configuration—Test the system under all combinations of allowable configurations.

Disaster Recovery—Mimic a disaster situation and follow the procedures.

Security—Attempt all possible vulnerabilities.

Usability—Is the system user-friendly? Give it to actual users and ask them.

Regression—Following the fix of a defect, make sure that a new defect wasn’t introduced.

Path or Branch—Test every possible path that the system can follow at least once.

Integration—Test that the different components of the system work together

User Acceptance—Final test executed upon delivery of the system by the users.

The Value of Testing

It is important to remember that thorough testing does not ensure a quality product. Testing does not do anything to the product itself. Testing is a vehicle for measuring the quality of the product. Testing produces reliable information grounded in the observed behavior of the system. The value of this information correlates to the sum of all the changes in our confidence in the system’s valuable behaviors. Testing is about confidence, which will change depending on the results of the testing. One of the important questions that a Project Manager must keep in mind during testing is "When to stop testing?" When additional testing adds too small a change in your confidence it is time to stop.

Defect Tracking

It is critical during the test phase to diligently record all defects found, and to account for all of them before deciding that testing is complete. Keep in mind that as much as 35% of all defects found are introduced as a result of fixes. The importance of reevaluating the system, module or work product following changes cannot be underestimated.

Change Control

Change control for all work products of the project is necessary to eliminate the risk of introducing new defects. Part of the high cost of fixing defects in the latter stages of the project are attributable to assessing the impact of a change on all the existing work products.

Summary

This paper has defined and scratched the surface of the basic concepts that are part of Quality Management. Project Managers need to manage the quality of their project from inception through the User Acceptance Test. The cost of Quality Assurance may be high. It is certain that the cost of low quality is much higher. When practiced and managed carefully, QA practices will save time and money, and will lead to a more mature and efficient project organization.

References


