Project Risk Management: Independent Software QA Ensures Success

Introduction

We hear the stories time after time: A group of talented, hard-working, motivated software engineers has once again produced a low-quality software product—late. This phenomenon has given birth to the "traditional" viewpoint that a project's quality and its schedule and/or cost must be traded off against each other. And it is this fundamental misconception that starts the downward spiral that results in too much time and money being spent on a project that was doomed to fail before it even got under way.

This is where independent software quality assurance (QA) services come in.

Quality assurance is a formal process with well-defined steps that is used to evaluate and document the quality of all work produced during each stage of the software development life cycle (SDLC). This process ensures that the customer's requirements are identified and standards are established and adhered to, starting with the planning phase of a software development project.

The independent QA team should not be perceived as conducting an "us versus them" witch hunt—it is not about a lack of confidence in the software developers. Instead, it is akin to taking out an insurance policy: Investing in an organization that tracks each stage of planning, development, and deployment against the end users' requirements ensures that errors and anomalies are detected and corrected early in the process, which in turn reduces costs and saves time.

QA Defined

The definition of QA in the Institute of Electrical and Electronics Engineers' (IEEE) *Handbook of Software Quality Assurance* states, "Software quality assurance is the set of systematic activities providing evidence of the ability of the software process to produce a software product that is fit to use." This definition ties the development process directly to the end goal—the process is not about software development, but rather, the usefulness of the software that is developed.

Other definitions of QA are less formal but no less true:

- Quality is "hard to define, impossible to measure, easy to recognize."*
- "Quality is generally transparent when present, but easily recognized in its absence."†

A system that has the concept of quality built in to it will work as and when the end users need it to work. If quality checks have not been included throughout the development of the system, the end product will contain errors and may even fail catastrophically when it is deployed.

QC and IV&V versus QA

Quality control (QC) occurs *as part of* the software development team's tasks—the same vendor or contractor who designs, develops, tests, and implements the system also performs the QC functions on the software. And because the same vendor or contractor is performing the QC functions, there is no one who can offer outside, “fresh eyes” oversight during the project.

IV&V (independent verification and validation) is formally defined as an engineering discipline that employs rigorous methods for evaluating the correctness and quality of the software product throughout the software development life cycle from a *system-level* point of view. Rather than being an integral part of the SDLC, it takes place side by side with software development, testing, and integration. IV&V is funded and managed by an entity that is outside the jurisdiction of project management, whereas independent QA efforts usually fall within the scope of the Project Management Office (PMO).

The issue that development teams face is that their members become deeply involved in the intricate, day-to-day challenges of the project, which, as part of human nature, makes it extremely difficult to pull back from the trees and really see the whole forest. That is the job of the QA team.

Characteristics of QA

Independence

The QA team comes under the same Project Management Office as the development team, but QA focuses on making sure that all development activities follow the standards and guidelines that it established during the planning phase so that the end users’ expectations are met. Independent QA functions as a set of checks and balances: The development team members are able to keep their eyes on the prize—a software system that works properly and meets mission requirements—knowing that an independent group is there to help keep the project on track in terms of time, budget, and resources.

Scalability

In addition to being independent from the QC performed by developers, the QA process is *scalable* and *flexible*. It is not a rigid methodology, but rather, one that can be tailored to any size project to address unique technical, functional, budgetary, and performance requirements in terms of the project’s size and complexity. A more complex project or system will have more phases that it must pass through, and thus there will be more critical junctures at which QA processes, standards, and procedures should be employed—compare, for example, a major upgrade to human resources systems modules in a large Cabinet-level federal agency to a new employment application tracking program at a small private business. QA can be used for both, but at radically different levels of effort.

Is QA Just a Trend?

Software quality assurance is an established, proven process. The International Organization for Standardization (ISO) and IEEE have published numerous handbooks and standards that serve as the industry-wide gold standards for including QA in software development projects.

Until the mid 1980s, most organizations published their own standards or codes for software development vendors to follow, and their staff would audit vendors regularly to make sure they followed those guidelines. It was not unusual for a single vendor to be audited separately by several different customers, each with its own quality system codes. Then, in 1987, ISO published a series of standards known as ISO 9000. The most recent standard that applies to software QA is ISO 9001:2008, *Quality Management Requirements*.

Starting around the same time, IEEE began publishing software QA standards and guides. In addition to the *Handbook of Software Quality Assurance*, other guides include the *IEEE Standard for Software Quality Assurance Plans* (no. 730-2002) and the *IEEE Standard Dictionary of Measures to Produce Reliable Software* (no. 982.1-2005).

The increasing complexity, size, and importance of software applications has led to a steadily increasing demand for independent QA.

Implementing QA

A QA team is needed to control the process and assure product quality through planned and as-needed QA activities. These activities include technical reviews, evaluations, and audits of all SDLC-required work products (technical documents and source code) and infrastructure (hardware, software, and tools).

Software project sponsors should consider establishing the QA process as early as the project planning phase so that the QA team members will have an unbiased, impartial view into project planning, scheduling, budgeting, and resource allocation.

Including the QA team at the beginning of the project ensures development team compliance with the mandated scope and functionality of the software. Early implementation of QA helps prevent cost overruns and schedule slippages by means of both preventive and corrective actions to remedy risks, gaps, and issues detected in early phases of the SDLC. This approach serves to increase developer productivity and customer satisfaction.

QA activities are performed during each SDLC phase to provide timely feedback to the software development team so that its members can develop and implement a quality software product. The major QA phases include

- planning,
- requirements analysis,
- design,
- development/coding,
- testing,
- deployment,
- training,
- maintenance and operation, and
- retirement.

SDLC in Brief

As its name implies, the software development life cycle is a continuous process, because any given software application is bound to require updates and maintenance during its lifetime.

As Figure 1 below shows, a new software development project starts with requirements analysis—a comprehensive survey and analysis of the functionality that the final software system must incorporate to satisfy users' needs. This functionality is usually documented in a requirements definition document.

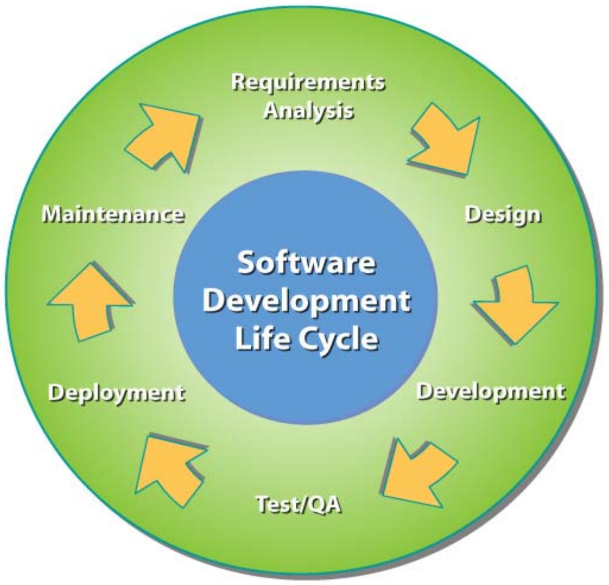

Figure 1. Software Development Life Cycle

The next step is to design the system based on the identified requirements. The design phase may include several increments, each one drilling down deeper into the details of each of the modules that comprise the system. A detailed design document is usually compiled at this stage, for use as a guide for the system developers.

The development stage is the one during which the code is written. For a system with more than one module, each module is coded individually.

During the test/QC phase (also called test/QA), individual modules are tested (*unit testing*). Once unit testing is complete and each module is shown to function as intended, the modules are combined into the complete system and tested (*integration testing*). The goal of integration testing is to ensure that there are no conflicts in the way the modules interact with each other. *System testing* shows whether the complete system behaves as expected. *Acceptance testing* is the final step to ensure that the system is ready to be placed into production for everyday use. Test cases are the most common tool for all types of testing, and detailed logs should be maintained to document all test results and any redesign or redevelopment effort that is required to fix errors.

Deployment is the stage at which the fully tested system is moved into production—that is, it becomes available for the end users.

Finally, any system is likely to need maintenance, which can include upgrades to existing functionality, creation of new reporting tools, and minor bug fixes.

The Role of QA alongside SDLC

The heart of a successful QA methodology is the quality check of the work produced during each phase of the SDLC, relative to itself and adjacent phases, against the standards and procedures that were established at the beginning of the project.

QA activities take place separate from, but in concert with, the SDLC stages, giving the QA team the chance to assess risks and suggest redirection at each stage in the process. Each QA phase is designed to determine whether development products of an SDLC activity conform to the requirements, specifications, and standards defined for that activity.

QA Phases

Table 1 maps the QA activities to the SDLC phases.

Table 1. QA Activities by SDLC Phase

SDLC Phase	QA Activities
Planning	The QA team monitors the project planning effort to help ensure that the plans, schedules, and cost estimates are of high quality, meet contractual requirements, and have the backing of the key project stakeholders.
Requirements Analysis	• *Work Product Evaluation*—The QA team reviews the Functional Requirements Documents (FRD) to help ensure that they meet the delivery requirements and quality standards. • *Process Audit*—The QA team evaluates whether the process that the development contractor uses to create the FRD complies with the approved plans.
Design Phase	• *Work Product Evaluation*—The QA team reviews the System Design Documents to help ensure that they meet the delivery requirements and quality standards. • *Process Audit*—The QA team evaluates whether the process that the development contractor uses to create the System Design Documents complies with the approved plans.
Development	• *Work Product Evaluation*—The QA team participates in the Code Walkthrough process to help ensure that the source code meets the coding standards. • *Process Audit*—The QA team evaluates whether the process that the development contractor uses to create the source code complies with the approved plans.

SDLC Phase	QA Activities
Test	• *Test Readiness Assessment*—The QA team determines whether the system is ready for the development contractor to conduct unit, subsystem, integration, and system tests. • *Test Monitoring*—The QA team monitors the tests performed by the development contractor to help ensure that these tests are executed as planned, that all test results are properly documented, and that all anomalies are appropriately resolved.
Deployment	• *Work Product Evaluation*—The QA team reviews the implementation and deployment plans and attends the Production Readiness Review meetings to help determine whether the system is ready to be deployed. • *Process Audit*—The QA team evaluates whether the process that the development contractor uses to deploy the system complies with the approved plans.

As part of the QA process during each phase of the SDLC, the QA team also documents and reports all issues that require corrective action. The team also continuously monitors these issues to help ensure that the corrective action appropriately resolves the issues.

If the QA analysis determines that the project scope, objectives, and goals are unrealistic and not implementable within the available resources and constraints, the team notifies and works closely with the PMO and users to minimize the risks and prevent project failure.

As early as the requirements definition phase, quality assurance analysis can prevent issues from arising because all nonfunctional and untestable requirements are detected in the technical reviews. As a result, these requirements are removed from the FRD and are replaced with ones that are functional and testable.

By the same token,

- system design errors are detected and fixed during the design review phase,
- coding errors are identified and corrected during the code inspection session, and
- system anomalies are detected and resolved during the independent testing phase.

Detecting defects and errors in the SDLC phases helps to prevent defective software from going into production.

Conclusion

Independent software quality assurance is a time-tested methodology that provides a relatively low-cost insurance policy at the outset of any software development project.

Having an independent QA team working together with developers can be critical to the successful development of programs, projects, and products and to the provision of services because its members provide independent oversight throughout the life of the project. This oversight ensures that the developers can meet all program plans, schedules, and deliverables with fewer hidden errors, issues, and anomalies, thus lowering the risk of unexpected cost and schedule overruns.

The QA team's findings will be unbiased and fair because its members have the same goals as project management—high-quality software that works as required, is developed on time and within budget, and meets all stakeholder requirements.

Futrend Technology, Inc.

As a Small Business Administration (SBA) certified 8(a) corporation, Futrend Technology, Inc., has positioned itself to assist federal clients with tasks ranging from systems engineering and development of Enterprise Resource Planning (ERP) solutions, software development, and system integration and implementation, to full life-cycle information technology support and professional services such as independent IV&V and QA, and certification and accreditation (C&A). We focus on maintaining our ability to "partner" with each of our customers to provide top-notch professional services and excellent results.

Futrend prides itself on excellent service delivery. Our corporate leadership focuses on providing quality services that meet or exceed expectations.

References

*Kitchenham, Barbara, and Shari Lawrence Pfleeger. "Software Quality: The Elusive Target." *IEEE Software* 13, 1, Jan. 1996: 12–21.

†Gillies, Alan C. *Software Quality, Theory and Management*. International Thomson Computer Press, 1997.