Chapter XVI
Software Quality Assurance

Dawn M. Owens
University of Nebraska at Omaha, USA

Deepak Khazanchi
University of Nebraska at Omaha, USA

ABSTRACT

Successful implementation of IT (information technology) projects is a critical strategic and competitive necessity for firms in all industrial sectors today. However, due to cost overruns, schedule delays, unfilled requirements and poor quality, it is reported that less than 30% of IT projects are perceived to be successful. Much has been written about causes of project failure and many have provided best practices and critical success factors for effective management projects, yet projects still continue to fail. As a first step to overcome systemic causes of project failure we propose a unified definition of software quality assurance (SQA). We use this definition to develop and present an approach to SQA that focuses on controlling risks and provide a framework for assuring the development and project management life cycles.

INTRODUCTION

Successful implementation of IT (information technology) projects is a critical strategic and competitive necessity for firms in all industrial sectors today. This is even more important in times of scarce resources needed for other competing strategic initiatives important to a firm. However, it is well established that a significant numbers of software and infrastructure projects still fail to deliver on time and within target costs and specifications (The Royal Academy of Engineering, 2004). Due to cost overruns, schedule delays, unfilled requirements and poor quality, less than 30% of IT projects are perceived to be successful (Glass, 2006).

Companies spend billions of dollars on IT projects each year, with the average spending at 4 to 5 percent of annual revenue on IT (Charette, 2005). In 2005, organizations spent an estimated $1 trillion on IT hardware, software, and services worldwide (Charette, 2005). Project failures have cost the US economy at least $25 billion and maybe as much as $75 billion (Charette, 2005). Based on the current project failure statistics, companies are faced with billions of wasted dollars on IT projects alone.

Many have written about the causes of project failure and have provided suggestions and critical success factors for effective management of projects. This chapter takes a different approach by examining how *software quality assurance practices* can impact project outcomes. It is the authors' contention that **software quality assurance (SQA)** plays a critical role in the software development lifecycle (SDLC) and can impact a project's overall success. Failure to pay attention to SQA can result in budget overruns, schedule delays, failure to meet project objectives and poor customer satisfaction (Chow, 1985). In fact, quality is considered a vital requirement of software products, a business essential, a competitive necessity, and even a survival issue for the software industry (Murugesan, 1994). Strong quality focus is emerging in all phases of the software development lifecycle with increasing emphasis on product quality, process maturity, and continual process improvements (Murugesan, 1994).

No matter how advanced the tools and techniques – all will come to nothing and the project will fail if the quality management system is not effective (Gill, 2005). The term SQA is often misunderstood, being viewed strictly in terms of software testing activities. However, in our view, SQA is a much broader concept. As we will present in this chapter, SQA is a set of assurance processes that are implemented across all phases of the project and software development lifecycles. Using this definition, we propose in this chapter a comprehensive SQA process. The SQA phases presented will provide IT professionals with a framework for software practitioners and managers to analyze their quality assurance practices and determine where to initiate changes that will reduce and/or prevent causes of project failure. The rest of the chapter is organized around four main objectives: (1) Discuss the reasons for project failure, (2) Define software quality assurance, (3) Propose a software quality assurance process and discuss how it can be used to assure quality, and (4) Discuss implications for research and practice.

REASONS FOR PROJECT MANAGEMENT FAILURES

The statistics on project management success are dismal and probably not surprising to most IT professionals. When a project fails it can jeopardize an organization's reputation with its customers and its name in the marketplace (Charette, 2005). In addition, project failures can be costly to organizations in terms of wasted dollars and resources. The bottom line is that projects continue to fail for many reasons.

It is generally accepted that software project failures predominantly occur due to cost or schedule overruns and quality control problems. Table 1 summarizes the most common reasons for project failure documented from published literature.

Many projects fail due to a combination of the factors listed in Table l (Charette, 2005). Complicating matters is the fact that there are a number of unique challenges related to software development itself, such as, technology, the development process, scope and complexity of projects, and risk of projects (Reel, 1999). However, since most causes of software failures are predictable and avoidable, premeditated planning and inclusion of proper controls and processes can result in diminishing of these concerns, resulting in project success. Project success requires carefully

Table 1. Common reasons for project failure

	Reasons for Failure	Citations
1	***Planning -*** Insufficient planning	[Nelson, 2007]; [Brown et al, 2007]; [Murugesan, 1994]
2	***Scope*** - Poorly defined requirements and scope	[Nelson, 2007]; [Charette, 2005]; [Baker, 2001]; [Reel, 1999]
3	***Stakeholder -*** Ineffective stakeholder management	[Nelson, 2007]
4	***Commitment -*** Lack of executive support or project sponsorship	[Nelson, 2007]; [Brown et al, 2007]; [Reel, 1999]; [Murugesan, 1994]
5	***Time -*** Poor estimation and/or scheduling; unrealistic deadlines	[Nelson, 2007]; [Galin, 2004]
6	***Cost -*** Unrealistic budget	[Galin, 2004]
7	***Quality and Testing***- Shortened or elimination of quality assurance practices; inadequate testing or reduction of testing time within the software development lifecycle;	[Nelson, 2007]; [Murugesan, 1994]
8	***Resources*** – Insufficient qualified resources	[Nelson, 2007]; [Galin, 2004]; [Charette, 2005]; [Murugesan, 1994] ; [Reel, 1999]
9	***Communication*** - Poor or ineffective communication	[Brown et al, 2007]; [Charette, 2005]; [Wong and Tein, 2004];
10	***Risk -*** Insufficient risk planning and risk management	[Nelson, 2007]; [Charette, 2005]
11	***Politics -*** Stakeholder politics; user politics; corporate politics	[Nelson, 2007]; [Charette, 2005]; [Reel, 1999]
12	***Stakeholder -*** Lack of user and stakeholder involvement	[Nelson, 2007]; [Brown et al, 2007]; [Murugesan, 1994]
13	***Project manager*** - Poor project management or project manager	[Charette, 2005]
14	***Process -*** Undefined development processes or lack of adherence to process standards (Sloppy development practices); failure to implement best practices and lessons learned	[Charette, 2005]; [Reel, 1999]
15	***Product -*** Speed to market, desire for innovation; deliver now and correct errors later	[Murugesan, 1994]
16.	***Project Goals -*** Unrealistic or unarticulated project goals	[Charette, 2005]; [Wong and Tein,2004]
17	***Change Control -*** Poorly managed change control	[Reel, 1999]
18	***Technology*** – changes in technology	[Reel, 1999]

followed processes that assure quality at every phase of the project and development lifecycles. Quality assurance practices can minimize project failures by providing checks throughout this process. Many people associate quality assurance with testing and most people think about testing once development is complete. However, developers and managers need to consider SQA and testing as being integral to every phase of the project lifecycle.

DEFINING SOFTWARE QUALITY ASSURANCE (SQA)

In this section we develop a unified definition for SQA. There are many different definitions of software quality assurance found in the literature (refer to Table 2).

These definitions are built on the premise that SQA is a process or set of activities that assures adherence to standards as well as assuring that the stated software requirements are met.

Table 2. Definitions of SQA

Definitions of Software Quality Assurance	
SQA is the planned and systematic approach to the evaluation of the quality of and adherence to software product standards, processes and procedures. It includes the process of assuring that standards and procedures are followed throughout the software lifecycle.	Agarwal et al., 2007
Quality assurance is providing assurance and credibility that the product works correctly.	Feldman, 2005
A set of actions necessary to provide confidence that the software development or maintenance process of a software system product conforms to established requirements while staying within the schedule and budget confines.	Galin, 2004
Software quality assurance is an umbrella activity applied to each step in the software process. It involves mapping managerial precepts and design disciplines of quality assurance onto the space of software engineering.	Gill, 2005
Conformance to functional and performance requirements, documented development standards, and implicit characteristics that are expected of all professionally developed software.	Pressman, 2005
A planned and systematic pattern of actions necessary to provide confidence that the product conforms to established requirements.	IEEE Std 730-1998
The purpose of SQA is to provide management with appropriate visibility into the software process being use and of the software product being built. It involves reviewing and auditing software products and activities to verify they conform to procedures and standards (Schulmeyer & McManus, 1999).	CMM
ISO9000 defines the requirements for a quality management system in order to produce higher quality products (Pressman, 2005). Quality is the totality of features and characteristics of a product that impact its ability to satisfy stated or implied needs (Stockman et al, 1990).	ISO9000

Another approach to defining SQA is to consider "*assurance as a set of services or a set of activities that inspire confidence and certainty*" (Merriam Webster, 2008). In this vein, Khazanchi and Sutton (2001) discuss assurance services and its impact on assuring quality specifically in business to business (B2B) electronic commerce related business practices. They define assurance services as a set of activities conducted by trusted, independent organizations to certify and/or validate business transactions between business partners by reviewing internal control mechanisms with the fundamental objectives to reduce risk, assess internal controls, and improve quality (Khazanchi & Sutton, 2001). Premeditated assurance against project failures can potentially be achieved through the establishment of controls. We think of controls as a *system of procedures, mechanisms or policies that could proactively prevent and detect software project failures*. A software quality assurance process should incorporate control mechanisms to assure conformity with an organization's policies, procedures, and standards.

Software quality assessment is difficult due to the uniqueness of software products (Rosqvist et al., 2003); therefore a well defined repeatable process is necessary in order to deal with the unique challenges. Repeatable processes such as CMMI have shown to result in higher levels of quality and project performance (Subramaniam, et al, 2007). Any such SQA process must also address schedule and budget constraints. Projects continue to fail because they do not meet stated time or cost constraints. It is important that the implementation of such a process does not negatively affect the ability to deliver on time and within budget.

Gill (2005), describes SQA as an umbrella activity that is applied to each step of the software process. We believe that this umbrella activity should apply not only to the software development lifecycle, but also the project management lifecycle. Based on the above discussion, we define software quality assurance as follows.

Software Quality Assurance (SQA)

SQA *is a well defined, repeatable process that is integrated with project management and the software development lifecycles to review internal control mechanisms and assure adherence to software standards and procedures. The objective of the process is to assure conformance to requirements, reduce risk, assess internal controls and improve quality while conforming to the stated schedule and budget constraints.*

As stated earlier, the SQA process needs to be well integrated with other lifecycle processes. Therefore, SQA cannot occur late in the lifecycle as a last attempt to add quality at the end of development activity (Voas, 2003). The next section provides a more in depth discussion of the SQA process.

THE SQA PROCESS AND ROLE OF SQA TEAMS

Many developers and managers continue to believe that quality is something to think about after the code has been compiled. Although this might be the reality, it should not be the practice. The SQA process is a *continuous assessment* mechanism deployed throughout the project and development lifecycles with specific controls and documentation requirements (Thayer & Fairley, 1999). The process should be used to control and assure the software development process, assure a quality outcome (e.g., zero defects) and assure project success (e.g., on-time within budget delivery). The activities in the SQA process may include formal reviews, testing, control of documentation, measurement, and reporting (Pressman, 2005). At a high level, the function of SQA is to perform the following (Galin, 2004):

- **Assure software project planning has taken place:** Quality practices should be planned well in advance so that there is time for them to be implemented.
- **Assure user requirements:** Requirements should be reviewed from beginning to end to assure conformance to established standards and conformance to user needs.
- **Assure the design process:** Provide guarantee that methodologies are followed and requirements are met by the design.
- **Assure coding practices:** Coding standards, practices, and guidelines must first be established, and then adhered to.
- **Assure software integration and testing has taken place:** Software integration and testing should be planned, implemented, and executed accordingly
- **Conduct random and scheduled audits:** Perform SQA audits to assure the necessary controls are in place.

The SQA process consists of a variety of phases with specific activities. These activities should be performed by a group such as an SQA team that is independent of the project participants. The software engineers should have responsibility for performing the technical work while the SQA group is responsible for software quality assurance planning, analysis, and reporting (Pressman, 2005). SQA is most effective when it reports up through a separate management team so they can remain committed to the process and remain objective to the deliverable. The teams should consist of appropriately skilled professionals (Godbole, 2004). The responsibilities of the SQA team include review of documentation (development plans, testing plans, project plan) for completeness and adherence to standards, participation in inspections, review of test results, and periodic audits of controls (Godbole, 2004). The SQA team should document any plans or processes that do not reflect adherence to standards and best practices. The SQA team should

also conduct all design reviews and audits that regulate the software development and project management activities.

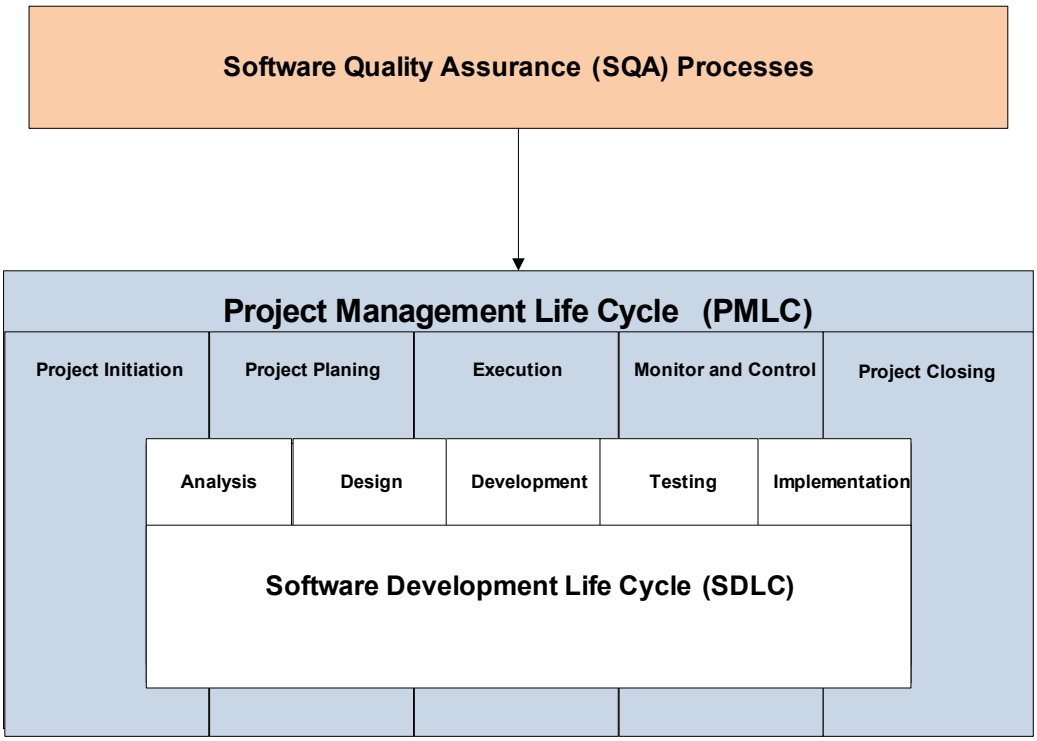
The functions performed by the SQA team are similar to that of IT auditors. Both are concerned with the existence of standards, adherence to standards, and documenting deviations (Weber, 1999). The existence of an SQA team can change how both internal and external auditors perform their work. SQA personnel are dedicated to quality assurance; therefore, the SQA team will likely undertake more comprehensive checking of controls than auditors (Weber, 1999). However, in the absence of an SQA team, auditors will perform many of the checks that the SQA team would perform.

Once a SQA process has been fully implemented it can help eliminate defects and prevent ambiguous and changing requirements resulting in improved customer satisfaction (Schulmeyer & McManus, 1999). The cost to find and repair defects or problems increases significantly as a project progresses through the requirements phase to the implementation phase. Implementing an integrated SQA process has the potential for generating cost savings in the long run.

THE SQA PROCESS EXPLORED

The SQA process has its own lifecycle and covers all facets of a software project. It typically requires careful planning and gathering of data in the form of various snapshots of artifacts (Feldman, 2005). For example the SQA plan, project management plan, and project metrics are potential artifacts that could be found in the project document repository. SQA is not a single

Figure 1. Software Quality Assurance (SQA) Process integrated with PMLC and SDLC



process, but an approach that provides assurance and credibility throughout the project management and development lifecycle processes (Feldman, 2005). Figure 1 represents the SQA process and shows how it encompasses the project management and development processes.

The combined project management life cycle (PMLC) and systems development life cycle (SDLC) consists of eight unique phases – initiation, planning, analysis, design, development, testing, implementation, and closing. For each one of these phases there is a corresponding SQA process – SQA initiation, SQA planning, requirements assurance, design assurance, development assurance, testing assurance, implementation assurance, and SQA closing. Each SQA phase contains a feedback loop to the corresponding PMLC or SDLC phase. The purpose of the feedback loop is to provide input regarding issues or problems found during SQA activities and to ensure continuous improvement. Figure 2 denotes each of the SQA phases aligned with each of the eight combined PMLC or SDLC phases and the feedback loop. Each SQA phase has specific outputs which provide quality controls and documentation throughout the life of the project.

SQA LIFECYCLE PHASES

The following section discusses each of the phases in the combined project and development lifecycles along with the corresponding SQA activity. We provide a formal definition of each phase and list the inputs, outputs, and controls associated with that phase. We will begin with the first phase – project initiation.

Project initiation (see Box 1) is a project management function. This phase is the official start or formal kickoff of a project. The project manager is assigned and they begin working on the necessary initiation steps. The corresponding SQA phase, SQA initiation is intended to notify the SQA team that a new project is being initiated. One of the key outcomes of this phase is to define the quality control and audit processes for the project and to ensure that the proper controls are in place.

- **Inputs**
  - Feasibility Analysis
  - Business Case Document
- **Outputs**
  - Project Charter
  - Preliminary Scope Statement
- **Controls**
  - Verify the appropriate outputs have been completed.
  - Check the content of the outputs for correctness, consistency, and completeness (3 Cs)[1].

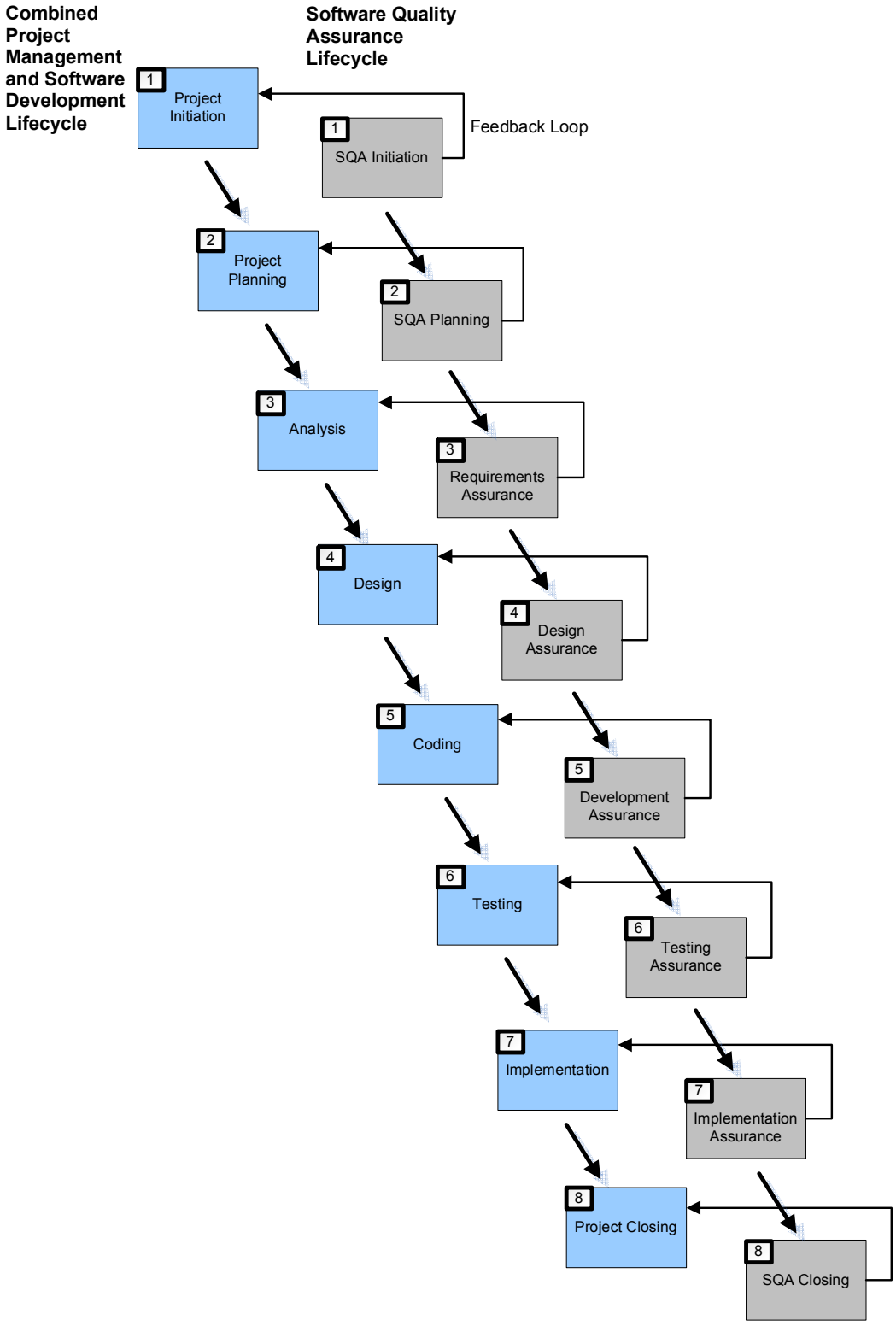
During project initiation the project charter is completed, which identifies the project sponsor, stakeholders, project objectives, and preliminary scope statement.

The controls in this phase include assuring that the proper outputs have been completed. The

Box 1. Project Initiation ⇒ SQA Initiation

Definition
Project Initiation: The launching of a process that can result in a new authorized project with a formal scope definition (PMBOK, 2004). **SQA Initiation:** The launching of SQA activities after a project has been authorized.

Figure 2. Combined PMLC, SDLC, and SQA process



project charter document should be verified for correctness, consistency, and completeness. For example, the following questions need to be addressed: Has the project manager been identified and have the key stakeholders been identified? Have the project team members been informed? Have the stakeholders been engaged? Have the development and SQA teams been engaged?

The potential risks at this phase of the project are low, however, if not identified can be detrimental to the overall project outcome. The major risk in this phase is the risk of initiating a project without a thorough feasibility analysis or business case. Without a feasibility analysis or business case, projects may be initiated without formal executive signoff, without clear alignment of strategic objectives, or without the necessary technological resources.

Project planning (see Box 2) is a project management function which involves deciding in advance what do, how to do it, when to do it, and who will do it. This phase includes those processes performed to define the scope, develop the project management plan, and identify the schedule and activities that occur within the project (PMBOK, 2004). It is during this phase that the team should agree upon which organizational processes will be used throughout the lifecycle. The planning phase should comprise 20-25% of the overall project timeline (PMBOK, 2004). SQA planning is important at this stage so that assurance practices can be successfully implemented (Godbole, 2004).

- **Inputs**
  - Project Charter
  - Standards and Procedures Manual
- **Outputs**
  - Project Management Plan
  - SQA Plan and Metrics
  - Project plans to represent the core knowledge areas (Project Scope Management Plan, Cost Management Plan, Communication Plan, Risk Management Plan, Procurement Plan)
  - Change management plan
  - Work Breakdown Structure (WBS) and WBS dictionary
  - Project Schedule with resource requirements
  - Roles and Responsibilities
- **Controls**
  - Verify the appropriate outputs have been completed.
  - Check the content of the outputs for the 3 Cs (correctness, consistency, and completeness).

The SQA plan needs to be developed in close coordination with the project management plan. The SQA plan documents the standards, practices, conventions and metrics to be used during the overall SQA process. It includes the reviews and audits that will be performed, the appropriate controls, documents to be produced by the SQA group, and the feedback to be provided to the software project team (Pressman, 2005). Stakeholders

Box 2. Project Planning ⇒ SQA Planning

Definition
Project Planning: defining the goals and objectives for a project and planning the course of action required to meet the scope and objectives. This includes specifying the processes and procedures that will be followed during the project. (PMBOK, 2004; Christensen & Thayer, 2001). **SQA Planning:** defining the goals and objectives of the software quality assurance plan which includes specifying any quality processes or procedures to be followed.

should review and approve both plans.

The primary function of the SQA team during this phase is to evaluate whether the nature and extent of planning are appropriate for the type of software being developed (Weber, 1999). The team must evaluate how well the planning work is being completed – for example, have resource requirements been accurately estimated (Weber, 1999). These assurance activities are conducted via interviews, observations, and reviews of documentation (Weber, 1999).

The potential risks at this phase of the project can be critical. A major challenge in many software development projects is incorrect schedules, incorrect cost estimates, inadequate project accountability procedures, and imprecise goals and success criteria – all products of the planning and analysis phases (Godbole, 2004). Not all of these risks can be prevented, but the key is to mitigate and control the risks by monitoring and ensuring that outputs include all facets that are relevant. For example, it is important to assure that the project schedule accounts for potential known risks.

Analysis (see Box 3) is an SDLC phase in which the project team gathers and documents formal user requirements. The software requirements are the foundation for subsequent phases (Pressman, 2005). The major steps in this phase include defining, reviewing, approving and base lining the requirements. Lack of confidence in requirements leads to lack of quality, therefore, it is important that the appropriate controls to ensure quality are implemented at this stage. During this phase the requirements are validated and the SQA team assures that the appropriate documents have been created.

- **Inputs**
  - Standards and Procedures Manual
- **Outputs**
  - Requirements Definition and Baseline
  - Requirements Traceability Matrix
  - Software Development Plan
  - High Level Design Plan
- **Controls**
  - Software Requirements Review
  - Preliminary Design Review
  - Verify the appropriate outputs have been completed.
  - Check the content of the outputs for the 3 Cs (correctness, consistency, and completeness).
  - Verify the requirements to ensure testability and feasibility.

Requirements verification includes a review of the requirements to ensure conformance to established standards. It also includes examining the specifications to ensure that all requirements have been stated unambiguously and to ensure testability (Pressman, 2005). If a requirement cannot be tested, it has not been stated clearly. After the requirements have been reviewed they should be approved by the major stakeholders. Once approved, a baseline is established. This baseline is used to track changes to the requirements in order to improve processes along the way

Box 3. Analysis ⇒ Requirements Assurance

Definition
Analysis: defining exactly what must be done to solve the problem by gathering and analyzing requirements and creating the initial software system model (Davis and Yen, 1999; Galin, 2004). **Requirements Assurance:** providing assurance that requirements are testable and complete.

and provide an audit trail of which requirements were changed.

Additional outputs in this phase include a requirements traceability matrix, a software development plan, and a high level design plan (Pressman, 2005; Galin 2004). A requirements traceability matrix is used in future phases to map the requirements back to specific functionality. It is a two-dimensional grid that lists out the requirements on one side and the functionalities on the other. This matrix is used in both development and testing. (Godbole, 2004 It can be used to determine how a change in one requirement will affect changes to other requirements or functionality. A software development plan determines how the software will be developed and includes a description of the processes and standards that will be used in development. A high level design (HLD) plan may also be created. According to IEEE standard 730.1 1995, the HLD should state which standards, practices, conventions and metrics will be used for specifying the internal structure of each program (IEEE Std730, 1998).

Due to its importance, the analysis phase of software development needs careful consideration in terms of risk. If the proper controls are not in place there is the potential for poor project outcomes. Incorrect, incomplete, unclear or inconsistent requirements can lead to risks such as poor customer satisfaction, not meeting customer expectations, or poor scope management. (Godbole, 2004). It is very important that the scope statement is managed and controlled in order to prevent scope creep, which can result in schedule delays or cost overruns. It is at this point that scope changes need to be documented, verified and approved.

In the design phase (see Box 4), the information from the analysis phase is translated into a blueprint that can be used for developing the software (Pressman, 2005). Lack of quality during design can invalidate good requirements (Godbole, 2004). During the design process, it is determined which standards will be used as well as the design process that the software team will use to perform the work (Pressman, 2005). Standards also address matters such as naming conventions and use of program design language (IEEE Std 730, 1998). If those standards are not followed, the end result is a product with poor quality (Pressman, 2005).

The objective of design assurance is to assure the processes being used to create the design conform to standards and that design is verified against the established requirements. Once a design process (i.e. waterfall, iterative, or prototyping) is chosen by the software design team, the SQA team reviews the selected design process in order to ensure compliance with organizational policies, internal software standards, or externally imposed standards such as ISO9000 (Pressman, 2005). Product and process metrics can be used to measure design coverage and complexity. Examples of process metrics used to measure the development process include development time and average level of developer experience

Box 4. Design ⇒ Design Assurance

Definition
Design: an iterative process where requirements are translated into an architecture or blueprint for developing the software and the inputs, outputs and processing procedures of the system are defined (Pressman, 2005; Galin, 2004).
Design Assurance: executing the appropriate controls to assure design has been completed according to stated policies and standards as well as assuring the necessary outputs have been completed.

(Godbole, 2004). Examples of product metrics include lines of code to measure size and number of integrated modules to measure complexity (Godbole, 2004). Quality metrics include lines of code and test case coverage (Godbole, 2004). It is important to identify metrics during the analysis and design phases to ensure that the proper measures are put in place.

- **Inputs**
  - Requirements Definition and Baseline
  - Software Development Plan
  - High Level Design Plan
  - Requirements Traceability Matrix
- **Outputs**
  - Detailed Design Plan and Design Process
  - Metrics Definitions to be used in the process (Product Metrics, Process Metrics, and Quality Metrics)
- **Controls**
  - Design walkthrough and inspection
  - Evaluation of the design process to ensure planned methodologies are followed
  - Check the design process for adherence to organizational policies and standards.
  - Verify the appropriate outputs have been completed.
  - Check the content of the outputs for the 3 Cs (correctness, consistency, and completeness).

SQA controls in this phase are implemented to assure adherence to the defined process and to assure a completed design. The SQA team is primarily concerned with determining whether the programmers used a systematic approach to the design of the software (Weber, 1999). For example, the team should utilize design practices that support a design plan and modular reusable code. The team can obtain evidence of design practices by conducting interviews, observations of programmers at work, and review of documentation (Weber, 1999).

Design walkthroughs and inspections are effective for providing traceability of requirements back to design, identifying potential defects, increasing programming quality, and increasing effectiveness of testing activities (Schulmeyer & McManus, 1999). Best practice suggests that for every hour of design a software engineer should spend .59 hours in design review and for every hour of coding they should spend an average of .64 hours of code review (Humphreys, 2004).

During development (see Box 5), design is translated into code. At this stage test plans and test cases are developed so that when the software is implemented testing can be done. The SQA team works on assuring that the design process is if high quality and that testing plans are appropriate for the implementation.

- **Inputs**
  - Standards and Procedures Manual
  - Software Development Plan
  - Detailed Design Plan

Box 5. Development ⇒ Development Assurance

Definition
Development: code is generated, reviewed, compiled and tested (Pressman, 2005).
Development Assurance: assure that the development team is following the stated development process and coding standards.

- **Outputs**
  - Code (testable software product)
  - Test Plan and Test Cases
  - Metrics
- **Controls**
  - Verify unit testing
  - Conduct random audits through code walkthroughs, inspections, and peer reviews
  - Audit Test Plan
  - Check the code to ensure adherence to standards
  - Verify the development process for adherence to organizational policies and standards.
  - Verify the appropriate outputs have been completed.
  - Check the content of the outputs for the 3 Cs (correctness, consistency, and completeness).

The objective of writing test cases is to detect undiscovered errors, not to show that the product functions (Godbole, 2004) correctly. As the testing team reviews the requirements and design documents to create the test plans, they should document any problems they encounter with the design. If the requirements document is incomplete or unavailable, the ability to detect defects is compromised (Baker, 2001). Therefore, this is one of the reasons why controls for complete requirements are necessary. It is much less expensive to catch these problems earlier, before the code has been completely developed.

The controls in this phase are designed to review the code to catch defects, even before testing begins and to capture metrics that were defined during the planning and design phases. The controls are also used to verify development processes and standards were followed. Unit testing and code walkthroughs are the two methods used to catch defects. The SQA team should evaluate how well unit testing has been performed by evaluating whether a systematic approach to unit testing has been chosen (Weber, 1999). Random code walkthroughs, inspections, and peer reviews are very effective tools to address the basic question whether the design has been correctly converted to the target coding language (Schulmeyer & McManus, 1999). Additional issues to consider are: Does the code meet the specified requirements? Have the programmers followed established programming conventions (Weber, 1999)? Is there any erroneous, ineffective, or inefficient code (Weber, 1999)? Random inspections conducted by the SQA team can help uncover defects early, before they become costly or before they become so deep-rooted in the system that it is impractical to remove them (Grady, 1993; Boehm, 2001).

One of the biggest risks is developing a product that does not meet the stated requirements or developing a product full of defects. The development controls and assurance activities are necessary to mitigate these risks and detect defects early and the risks associated with software development are diminished.

The goal of testing in SDLC (see Box 6) is to find and document defects. A defect is something

Box 6. Testing ⇒ Testing Assurance

Definition
Testing: the goal is to uncover as many software errors as possible in order to achieve an acceptable level of quality assurance (Galin, 2004). **Testing Assurance:** providing assurance that adequate testing has been completed and defects have been tracked and recorded.

that detracts from the products ability to meet the stated requirements (Humphreys, 2004). Testing activities include integration testing, automated testing, regression testing, and performance testing. The SQA team needs to work closely with the testers and developers to ensure testing procedures are of the highest quality (Weber, 1999).

- **Inputs**
  - Test Plan and Test Cases
- **Outputs**
  - Test Summary Report
  - Implementation Plan
  - Metrics
- **Controls**
  - Review Test Summary Report
  - Review Testing procedures (Test data, test environment, automated testing)
  - Review Implementation Plan
  - Verify the appropriate outputs have been completed.
  - Check the content of the outputs for the 3 Cs (correctness, consistency, and completeness).

The foundations of software testing include defining test processes, test cases and test plans, testing techniques, methodologies, tools, standards, and testers (Murugesan, 1994). Once testing is completed, the test results should be documented in a test summary report and reviewed with the developers and stakeholders. This important step is necessary before making a decision on whether or not to proceed with implementation. The test summary report should also provide a list of the documented risks of moving forward with the known defects.

The purpose of the controls in this phase is to review the testing results to ensure complete and rigorous testing procedures were followed and to assure metrics have been captured. Metrics are important for measuring performance and to provide useful measures for future projects. If these controls are not implemented correctly, there is risk in deploying a product that has many defects.

Once the project team has been given the okay to move forward, the implementation team (see Box 7) will install the product into production. The implementation team should carefully review the implementation plan including the change management and conversion strategy. The SQA team should conduct a post release review and ensure that production testing of the software is complete and satisfactory to the users.

- **Inputs**
  - Implementation Plan
- **Outputs**
  - Implemented Software
- **Controls**
  - Product Release Review
  - Verify the appropriate outputs have been completed.
  - Check the content of the outputs for the 3 Cs (correctness, consistency, and completeness).

Box 7. Implementation ⇒ Implementation Assurance

Definition
Implementation: after the software system has been approved, the software is installed into the target production environment (Galin, 2004). **Implementation Assurance:** providing assurance that the necessary implementation steps have been completed prior to and after implementation.

The primary control in this phase is the product release review. The product release review is necessary in order to assure there are no new defects present once the software is in production.

Both the project team and the SQA team should wrap up any project activities together including finalizing the project documentation and notifying the stakeholders and sponsors of project completion (see Box 8).

- **Inputs**
  - Metrics
  - Documentation from prior phases
- **Outputs**
  - SQA Summary Report
  - Post Project Review (lessons learned)
  - Metrics
- **Controls**
  - Verify the appropriate outputs have been completed and delivered.
  - Check the content of the outputs for the 3 Cs (correctness, consistency, and completeness).

During this phase there should be a formal documentation of lessons learned through a post project review. It is necessary to take the time to understand what went right and what went wrong so as not to repeat the same mistakes. It is also important to gather all the metrics that have been collected during the project. These metrics can help identify trends and best practices for future projects.

IMPLICATIONS FOR PRACTICE

Based on our experience and anecdotal evidence, we believe that using a well defined SQA process can substantially improve the chance of project success. Our proposed SQA framework provides an overarching framework for practitioners to develop an integrated assurance process to mitigate potential risks associated with project management and software development.

Implementing a SQA process is not easy. It takes commitment and executive support. The term process has a negative reputation in the world of IT development. People think of a process as slowing down the delivery of software and as a result there is a reluctance to consider new processes. However, once the value of SQA has been shown, IT personnel are more likely to embrace it. Clearly, any SQA process needs to include metrics and concrete deliverables in order to measure its value and help provide support for the future improved development of software products.

Project managers educated about the importance of the SQA process can champion its inclusion at the beginning of a project and not as an afterthought. Clear communication on the expectation of SQA is also important. All stakeholders need to understand and agree upon the role of the SQA process and team.

Organizations should have a dedicated SQA staff. This staff needs to be educated on risks, organizational standards, and policies and procedures relating to software development. Quality should be part of everyone's job and not just the

Box 8. Project Closing ⇒ SQA Closing

Definition
Project Closing: the processes performed to end all activities of a project, transfer the completed project, or close a cancelled project (PMBOK, 2004). **SQA Closing:** completing all the assurance process which includes making sure the necessary project closing activities have been completed.

responsibility of top management (Gill, 2005). Having management support for implementing SQA processes in all software development projects can go a long way to ameliorating risks and reducing software defects.

IMPLICATIONS FOR RESEARCH

Clearly there is consensus from research about why projects fail. However, little effort has been made in proposing processes that can integrate development with software quality assurance. This chapter proposes one approach for achieving this goal. Further research needs to be conducted on whether incorporating such a process leads to greater project success. In addition, research needs to be conducted to better understand the nature of risks in each phase of software development and the types of controls necessary. Finally, it would be also interesting to study how such controls are used and whether other factors need to be included.

CONCLUSION

In this chapter, we have discussed the reasons why projects fail and made a case for why a software quality assurance process is necessary to mitigate risk of failure and reduce failure rates. We have proposed a consolidated definition of SQA and developed a process for assuring software quality that encompasses the whole software development and project management life cycles.

REFERENCES

Agarwal, R. Nayak, P., Manickam, M., Suresh, P., & Modi, N. (2007). Virtual quality assurance facilitation model. *IEEE International Conference on Global Software Engineering (ICGSE 2007),* IEEE Computer Society.

Baker, E. R. (2001). Which way, SQA? *IEEE Software,* January/February 2001, 16-18.

Boehm, B., & Basili, V. R. (2001). Software defect reduction top 10 list. *Computer*, *34*(1), 135-137.

Brown, S. A., Chervany, N., L., & Reinicke, B. A. (2007). What matters when introducing new information technology. *Communications of the ACM, 50*(9), 91-96

Charette, R. N. (2005). Why software fails. *IEEE Spectrum,* September, 2005, *42*.

Chow, T. W. (1985). *Software quality assurance: A practical approach*. Silver Spring, MD: IEEE Computer Society Press.

Christensen, M. J., & Thayer, R. H. (2001). *The project manager's guide to software engineering best practices.* Los Alamitos, CA: IEEE Computer Society.

Davis, W. S., & Yen, D. C. (1999). *The information system consultant's handbook: System analysis and design*. Boca Raton, FL: CRC Press.

Feldman, S. (2005). Quality Assurance: Much More than Testing. *ACM Queue,* February 2005, (pp. 27-29).

Galin, D. (2004). *Software quality assurance: From theory to implementation*. Harlow, UK: Pearson Education Limited.

Gill, N. S. (2005). Factors affecting effective software quality management revisited. *ACM Sigsoft Software Engineering Notes, 30*(2), 1-4.

Godbole, N. S. (2004). *Software quality assurance: Principles and practice.* Pnagbourne, U.K.: Alpha Science.

Grady, R. B. (1993). Practical results from measuring software quality. *Communications of the ACM, 36*(11), 62-68.

Humphreys, W. S. (2004). *The software quality profile,* Retrieved September 12, 2004, from http://

www.sei.cmu.edu/publications/articles/quality-profile/index.html

IEEE Std730 (1998). IEEE standard for software quality assurance plans Std 730-1998. *IEEE standards software engineering,* Vol. Two, Process Standards, 1999 Edition, IEEE, (pp. 1-16).

IEEE Std1012 (1998). IEEE standard for software verification and validation. IEEE standards Software Engineering, Vol. Two, Process Standards, 1999 Edition, IEEE

Khazanchi, D., & Sutton, S. (2001). Assurance services for business-to-business electronic commerce: A framework in implications. *Journal of the Association for Information Systems, 1*(11).

Merriam Webster (2008). *Merriam Webster online: Assurance*. Retrieved March 17, 2008, from http://www.merriam-webster.com/dictionary/assurance

Murugesan, S. (1994, Dec. 21-22). Attitude towards testing: A key contributor to software quality. *IEEE's Proceedings of 1st International Conference on Software Testing, Reliability and Quality Assurance,* (pp. 111-115).

Nelson, R. R. (2007). IT project management: Infamous failures, classic mistakes, and best practices, *MIS Quarterly Executive, 6*(2), June 2007, 67-78.

PMBOK (3rd Ed) (2004). *A guide to the project management body of knowledge*. Newton Square, PA: Project Management Institute.

Pressman, R. S. (6th Ed.) (2005). *Software engineering: A practitioner's approach,* Boston: McGraw Hill.

Rosqvist, T., Koskela, M., & Harju, H. (2003). Software quality evaluation based on expert judgment. *Software Quality Journal, 11*(1), 39-55.

The Royal Academy of Engineering (2004). The challenges of complex IT projects. *The Royal Academy of Engineering and the British Computer Society*. Retrieved April 08,2008, from www.bcs.org/server.php?show=conWebDoc.1167

Reel, J. S. (1999). Critical success factors in software projects. *IEEE Software*, May/June, 18-23.

Schulmeyer, G. G., & McManus, J. I. (3rd Ed) (1999). *Handbook of software quality assurance*. Upper Saddle River, NJ: Prentice Hall.

Glass, R. L. (2006). The Standish report: Does it really describe a software crisis? *Communications of the ACM, 49*(8), 15-16.

Stockman, S. G., Todd, A. R, & Robinson, G. A. (1990). A framework for software quality measurement. *IEEE Journal on Selected Areas in Communications, 8*(2), February, 224-233.

Subramaniam, G. H., Jiang, J., & Klein, G. (2006). Software quality and IS project performance improvements from software development process maturity and IS implementation strategies. *The Journal of Systems and Software, 80*, 616-627.

Thayer, R. H., & Fairley, R. E. (2nd Ed.) (1999). software engineering project management: The silver bullets of software engineering. *Software Engineering Project Management*, (pp. 503-504). Los Alamitos, CA: IEEE Computer Society,.

Voas, J. (2003). Assuring software quality assurance. *IEEE Software,* May/June, 48-49.

Weber, R. (1999). *Information systems control and audit*. Upper Saddle River, NJ: Prentice-Hall, Inc.

Wong, B., & Tein, D. (2004). Critical success factors for enterprise resource planning projects. *Journal of the Australian Institute of Project Management, 24*(1), 28-31.

KEY TERMS AND DEFINITIONS

Design Assurance: Executing the appropriate controls to assure design has been completed according to stated policies and standards as well as assuring the necessary outputs have been completed.

Development Assurance: Assure that the development team is following the stated development process and coding standards.

Implementation Assurance: Providing assurance that the necessary implementation steps have been completed prior to and after implementation.

Requirements Assurance: Providing assurance that requirements are testable and complete.

Software Quality Assurance: A well defined, repeatable process that is integrated with the project management and the software development lifecycles to review internal control mechanisms and assure adherence to software standards and procedures. The objective of the process is to assure conformance to requirements, reduce risk, assess internal controls and improve quality while conforming to the stated schedule and budget constraints.

SQA Closing: Completing all the assurance process which includes making sure the necessary project closing activities have been completed.

SQA Initiation: The launching of SQA activities after a project has been authorized.

SQA Planning: Defining the goals and objectives of the software quality assurance plan which includes specifying any quality processes or procedures to be followed.

Testing Assurance: Providing assurance that adequate testing has been completed and defects have been tracked and recorded.

ENDNOTE

[1] IEEE Std 1012-1998 (IEEE Standard for Software Verification and Validation) uses the 3 Cs for V&V (Verification and Validation). The 3 Cs consist of correctness, consistency, and completeness.

APPENDIX A: SQA CHECKLISTS

PMLC and SDLC Phases	SQA Lifecycle Checklist Inputs	Outputs	Controls
1. Project Initiation	☐ Feasibility Analysis ☐ Business Case	☐ Project Charter ☐ Scope Statement	☐ Verification ☐ Check 3 Cs
1. Project Planning	☐ Project Charter ☐ Standards/Procedures Manual	☐ PM Plan ☐ SQA Plan & Metrics ☐ PM plans for core knowledge areas ☐ CM Plan ☐WBS & dictionary ☐ Project Schedule & resource requirements ☐ Roles & responsibilities	☐ Verification ☐ Check 3 Cs
2. Analysis	☐ Standards/Procedures Manual	☐ Requirements definition & baseline ☐ Requirements traceability matrix ☐ Software development plan ☐ High level design plan	☐ Software requirements review ☐ Preliminary design review ☐ Verification ☐ Check 3 Cs ☐ Verify for testability and feasibility
3. Design	☐ Requirements definition & baseline ☐Software development plan ☐ High level design plan ☐ Requirements traceability matrix	☐ Detailed design plan and design process ☐ Metrics	☐ Design walkthrough and inspection ☐ Evaluation of processes ☐ Check for adherence to process ☐ Verification ☐ Check 3 Cs
4. Development	☐ Standards/Procedures Manual ☐ Software development plan ☐ Detailed design plan	☐ Code ☐ Test plan & test cases ☐ Metrics	☐ Unit testing ☐ Code walkthroughs, inspections, and peer reviews ☐ Check code for adherence to process ☐ Verification ☐ Check 3 Cs
5. Testing	☐ Test plan & test cases	☐ Test summary report ☐ Implementation Plan ☐ Metrics	☐ Review test summary report ☐ Review testing procedures ☐ Review implementation plan ☐ Verification ☐ Check 3 Cs
6. Implementation	☐ Implementation Plan	☐ Implemented software	☐ Product release review ☐ Verification ☐ Check 3 Cs
7. Project Closing	☐Metrics ☐ Documentation from prior phases	☐ SQA summary report ☐ Post project review (lessons learned) ☐ Metrics	☐ Verification ☐ Check 3 Cs