

# Project Quality Management

July 1, 2011

CyberData Technologies, Inc.  
455 Springpark Place, Suite 300  
Herndon, VA 20170

## Document History

<b>Date</b>	<b>Version</b>	<b>Description</b>	<b>Author</b>
Feb 1, 2010	1.0	Initial Release	CyberData Technologies, Inc.
July 2011	1.1	Minor Edits	CyberData Technologies, Inc.

Copyright © 2010 CyberData Technologies, Inc. All rights reserved.

## Table of Contents

Overview.....	1
Method Activities.....	9
PQM 1.1 Define Quality Management Plan .....	10
PQM 1.9 Execute Definition Phase Deliverable Review .....	15
PQM 2.1 Prepare System Test Plan.....	19
PQM 2.9 Execute Requirements Phase Deliverable Review .....	25
PQM 3.1 Prepare Quality Architecture Design.....	29
PQM 3.2 Prepare Testing Environment Design .....	31
PQM 3.9 Execute High-Level Design Phase Deliverable Review .....	33
PQM 4.1 Define Functional Unit Test Cases.....	37
PQM 4.2 Define Integration Test Cases .....	39
PQM 4.3 Define System Test Cases .....	42
PQM 4.4 Define User Acceptance Test (UAT) Cases.....	45
PQM 4.5 Define Performance Test Cases .....	47
PQM 4.6 Install and Configure Testing Environment .....	49
PQM 4.9 Execute Low-Level Design Phase Deliverable Review .....	51
PQM 5.1 Educate IV&V Team.....	55
PQM 5.2 Define Test Scripts .....	57
PQM 5.3 Develop Automated Test Scripts .....	59
PQM 5.9 Execute Development/Construction Phase Deliverable Review.....	61
PQM 6.1 Execute Unit Testing.....	65
PQM 6.2 Build Release Candidate .....	68
PQM 6.3 Update IV&V Team.....	73
PQM 6.4 Execute Integration Testing .....	75
PQM 6.5 Execute System Testing .....	78
PQM 6.6 Execute User Acceptance Testing.....	81
PQM 6.7 Execute Performance Testing.....	84
PQM 6.9 Execute Testing Phase Deliverable Review.....	86
PQM 7.2 Educate Production Team .....	89

PQM 7.9	Execute IV&V Phase Deliverable Review .....	91
PQM 8.1	Prepare Production Build .....	95
PQM 8.2	Support Production Installation .....	97
PQM 8.9	Execute Deployment Phase Deliverable Review .....	99
Appendix – Sections to be Added .....		102
Appendix – Testing Glossary .....		103
Appendix – Sample ETL Unit Test Cases .....		112
Appendix – Kaizen Events .....		118
Appendix – Quality Management Plan Template.....		121
Appendix – Sample Quality Management Plan .....		122
Appendix – System Test Plan Template.....		136

## Table of Figures

Figure 1 – CyberData's Quality Management System .....	2
Figure 2 – Development and Testing V-Model .....	3
Figure 3 – Project Quality Management Activities by Phase.....	4
Figure 4 – Quality Management Deliverables by Project Phase .....	6
Figure 5 – CyberData Requirements Engineering Process .....	22

## Overview

### Why is Project Quality important?

According to the 2009 U.S. Federal Budget (OMB), 66% of all Federal IT dollars are invested in projects that are “at risk”.

Roger Sessions, a noted author and expert on complexity, developed a model for calculating the total global cost of IT failure. Roger describes his approach in a white paper titled [\*The IT Complexity Crisis: Danger and Opportunity\*](#). He concludes that IT failure costs the global economy a staggering \$6.2 trillion per year.

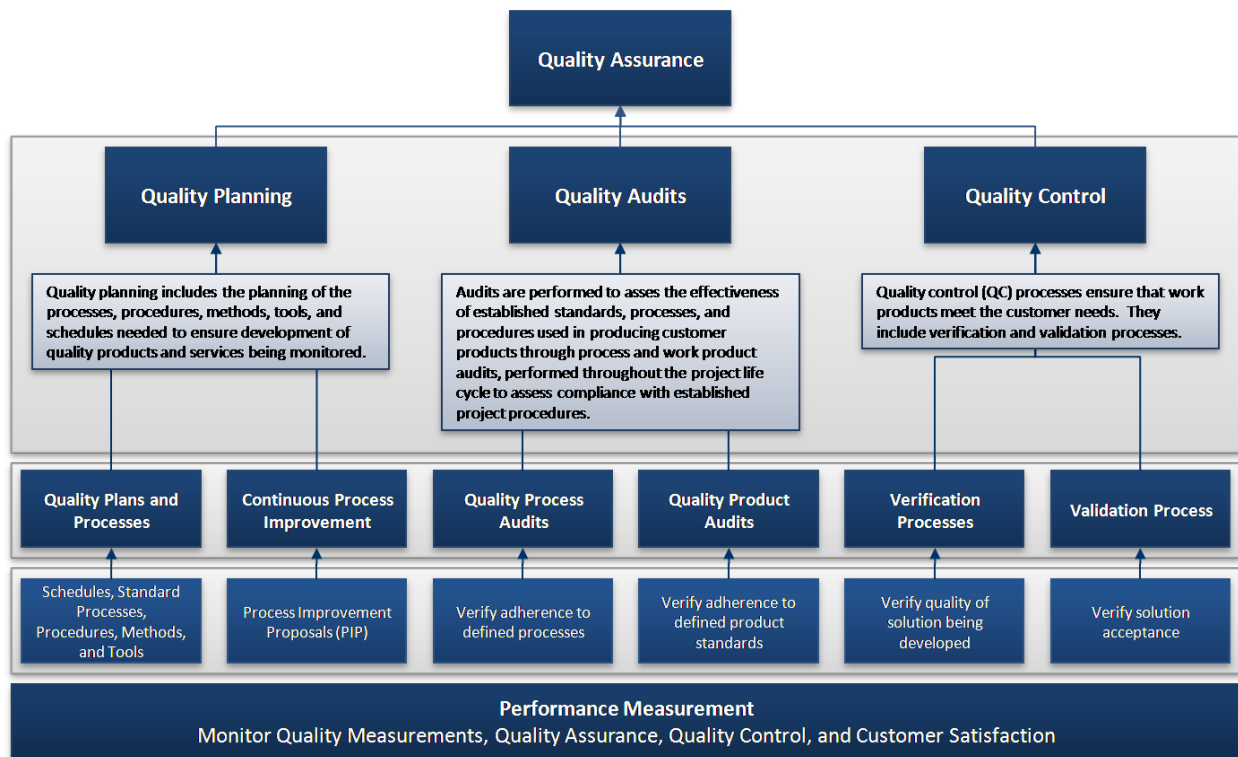
Quality is an encompassing term comprising utility, objectivity, and integrity. It is a measure of excellence, or a state of being free from defects, deficiencies, and significant variations. In some cases, we consider a product as “high quality” when it exceeds our expectations in several respects.

Quality is conformance to requirements or program specification; related to:

- Reliability
- Correctness
- Consistency
- Completeness
- Absence of bugs/defects
- Fault-tolerance
- Security
- Maintainability
- Documentation

At CyberData Technologies, quality management is integral to every facet of work performance. It begins with contract level planning and continues beyond contract completion through the integration of lessons learned into the next generation of work. Each CyberData employee has the responsibility for the quality of his or her work, and we provide incentives for initiating support to others that enable our teams to meet and exceed the quality goals of our customers and management.

CyberData’s Quality Management System helps us drive consistently high-quality performance across all our contracts and projects.



**Figure 1 – CyberData's Quality Management System**

CyberData's Project Quality Management (PQM) method is a key component of the Quality Management System. It emphasizes work product quality throughout the project lifecycle. CyberData's Project Quality Management, based on Total Quality Management (TQM) and Lean Six Sigma practices, integrates quality assurance and quality control procedures in each phase of a project. Each phase defines the expectations for the phase deliverables, verifies that the phase deliverables meet those expectations, and confirms that the project deliverables reasonably describe what needs to be done in the remaining phases of the project to meet client objectives.

CyberData's Project Quality Management method supports the industry-standard V-Model, depicted in the diagram below, which is applied to traditional waterfall and iterative project methods. Extreme and Agile methods will be addressed in an upcoming version of PQM.

Two project quality management activities are performed in each phase:

- Quality review of the normal project work products and deliverables
- Preparation and approval of deliverables that specifically focus on Project Quality Management

This method describes how projects are to conduct both types of activities. As the project progresses from definition to requirements to design activities, phase work products are reviewed and evaluated for their ability to accurately describe what needs to be done in

downstream phases. As the project progresses beyond development activities, quality management activities verify that development deliverables perform as specified and designed.

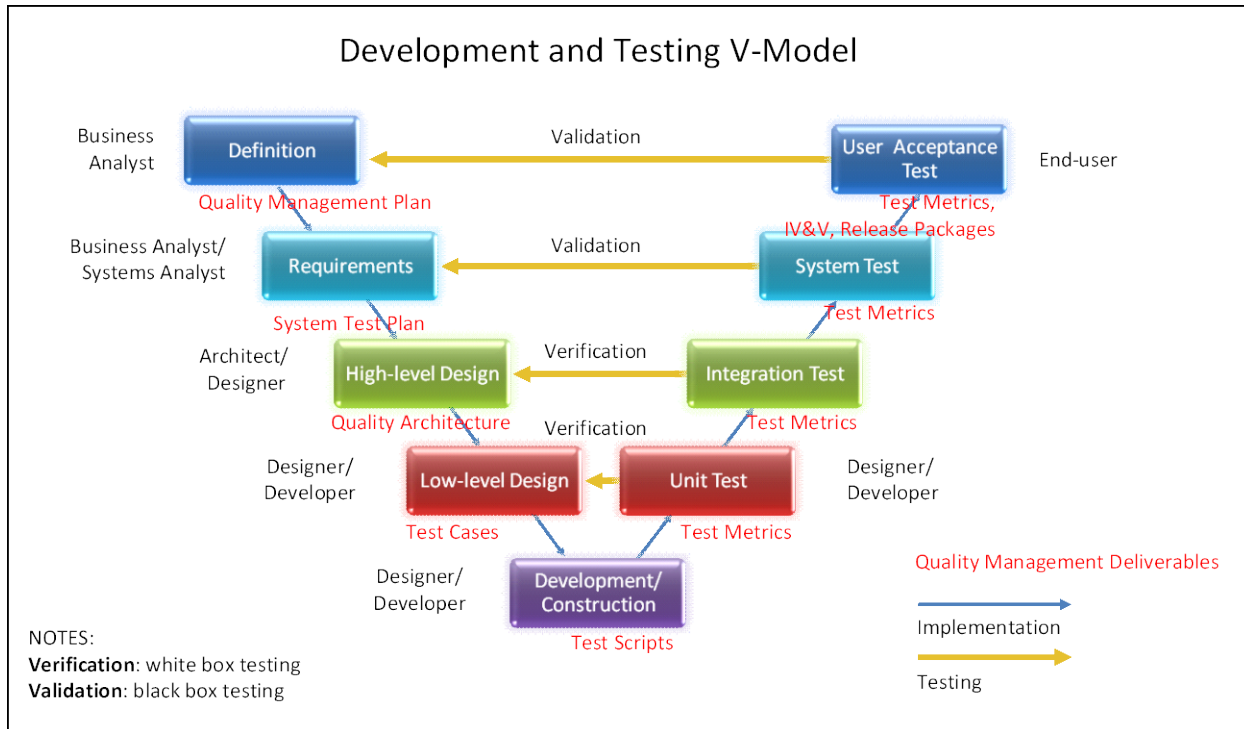


Figure 2 – Development and Testing V-Model

## Project Phases and Activities

The Project Quality Management method is divided into project phases, activities (highlighted in Figure 2 below), tasks, work products, and deliverables:

- **Project Phase** – A grouping of activities that produce a set of deliverables that will support the project phase that follows.
- **Activity** – A collection of related tasks that produce a deliverable.
- **Task** – A discrete work unit that contributes to the production of a deliverable, including producing a component of a deliverable. For example, some tasks produce individual sections of a document deliverable.
- **Work Product** – A tangible, physical result of a project activity or task.
- **Deliverable** – A work product that is named in the project contract, often described with specific contents and delivery date

DEFINITION	REQUIREMENTS	HIGH-LEVEL DESIGN	LOW-LEVEL DESIGN	CONSTRUCTION	TESTING	IV&V	DEPLOYMENT	USE
PQM 1.1 Define Quality Management Plan	PQM 2.1 Prepare System Test Plan	PQM 3.1 Prepare Quality Architecture Design	PQM 4.1 Define Functional Unit Test Cases	PQM 5.1 Educate IV&V Team	PQM 6.1 Execute Unit Testing	PQM 7.1 Prepare IV&V Build	PQM 8.1 Prepare Production Build	
		PQM 3.2 Prepare Testing Environment Design	PQM 4.2 Define Integration Test Cases	PQM 5.2 Define Test Scripts	PQM 6.2 Build Release Candidate	PQM 7.2 Educate Production Team	PQM 8.2 Support Production Installation	
		PQM 3.3 Review Project with IV&V Team	PQM 4.3 Define System Test Cases	PQM 5.3 Define Automated Test Scripts	PQM 6.3 Update IV&V Team	PQM 7.3 Support IV&V Testing		
			PQM 4.4 Define User Acceptance Test Cases		PQM 6.4 Execute Integration Testing	PQM 7.4 Review and Respond to IV&V Report		
			PQM 4.5 Define Performance Test Cases		PQM 6.5 Execute System Testing			
			PQM 4.6 Install and Configure Testing Environment		PQM 6.6 Execute User Acceptance Testing			
					PQM 6.7 Execute Performance Testing			
PQM 1.9 Execute Definition Phase Deliverable Review	PQM 2.9 Execute Requirements Phase Deliverable Review	PQM 3.9 Execute High-Level Design Phase Deliverable Review	PQM 4.9 Execute Low-Level Design Phase Deliverable Review	PQM 5.9 Execute Construction Phase Deliverable Review	PQM 6.9 Execute Testing Phase Deliverable Review	PQM 7.9 Execute IV&V Phase Deliverable Review	PQM 8.9 Execute Deployment Phase Deliverable Review	

Figure 3 – Project Quality Management Activities by Phase

### Activity and Deliverable Responsibilities

Each project quality management activity typically has multiple responsibility assignments, described in a responsibility assignment matrix. The responsibility assignment matrix is commonly known as a **RASCI matrix**. RASCI, pronounced “racy”, is an acronym derived from the five key responsibilities most typically used:

- **Responsible** – Those who lead the work efforts to achieve the task. There is typically one role with a participation type of *Responsible*, although others can be delegated to support in the work required.
- **Approver** – Those who must sign off (*Approve*) on work that *Responsible* provides.
- **Supporting** – Those who participate or assist in the achievement of the task
- **Consulted** – Those whose opinions are sought; and with whom there is two-way communication.
- **Informed** – Those who are kept up-to-date on progress, often only on completion of the activity or deliverable; and with whom there is just one-way communication.

### Quality Management Deliverables

Quality Management participates in the review and completion of all project deliverables, including communications, such as status reporting discussed in the previous section. The following table summarizes the major Quality Management deliverables to be produced by the project:

Project Phase	Quality Management Deliverable
Definition	Quality Management Plan
Requirements	System Test Plan
High-Level / Architecture Design	Quality Architecture Design
	Testing Environment Design
Low-Level / Application Program Design	Functional Unit Test Cases
	Integration Test Cases
	System Test Cases
	Performance Test Cases
	User Acceptance Test Cases
	Updated Requirements Traceability Matrix
	Testing Environment
Development / Construction	IV&V Presentation
	Test Scripts
	Automated Test Scripts
Testing	Build Plan
	Release Candidate
	IV&V Final Presentation
	Integration Testing Results
	System Testing Results
	Performance Testing Results
	User Acceptance Testing Results

Project Phase	Quality Management Deliverable
IV&V	Production Team Presentation

Figure 4 – Quality Management Deliverables by Project Phase

Each project phase produces a **Deliverable Statement of Intent (DSOI)** for each deliverable to be produced and approved in the phase. The DSOI describes the planned deliverable in sufficient detail that the client is able to evaluate the planned organization and content of the deliverable, and comment on the suitability of the planned deliverable. Achieving this agreement early in the phase is critical to deliverable acceptance later in the phase.

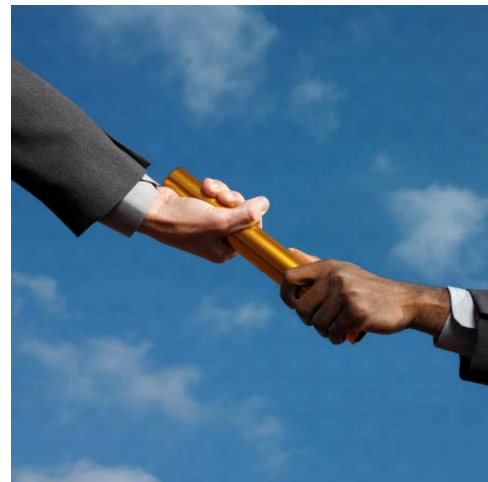
**Lessons Learned** are documented and collected at the end of each project phase, highlighting activities in need of adjustment or improvement to meet project objectives. Lessons Learned are a critical input into Kaizen Events – focused, short-term activities to improve a business process. Executed separately from the project, Kaizen Events include training followed by an analysis, design, and, often, re-arrangement of a business process. They are an important component of a continuous improvement program.

### Exit Criteria and Phase Deliverable Reviews

Each project phase should begin with the resources needed to effectively perform that phase. These resources include the completed and approved deliverables of the prior phase. Proceeding to the next phase before completing the current phase’s deliverables could leave the project team underprepared for the upcoming activities that depend on those deliverables.

Exit Criteria are established as integrated project phase fitness measurements that must be achieved before proceeding to the next phase. Exit criteria are defined and established in the methods that will be engaged by projects, as well as the methods used by the client and those specified in the project contract.

On an exception basis, as documented and approved (in writing) by the CyberData President and the client Project Manager, the Project Quality Management process can permit advancement to the next phase without completion of some exit criteria, but will condition that advancement on specific required actions and due dates to satisfy the exit criteria at the earliest possible date.



Before Exit Criteria are reviewed, during the Phase Deliverable Review activities, the Project Manager will verify that the set of deliverables for the phase meet contract requirements, are complete, and are considered acceptable for client review and approval.

As with all project activities, the Project Manager is responsible for scheduling sufficient time for Phase Deliverable Review activities, as well as any client review activities, prior to submission of deliverable to meet the contract's deliverable schedule.

## Key Considerations and Task Guidance

These topics capture advice and recommendations from experts' experience in conducting the activity and its constituent tasks.

## Project-Specific Quality Management Activities and Deliverables

The main body of this document describes the activities and deliverables that are standard and required across all CyberData projects. Responsibility assignments are listed for each activity and deliverable. Client-specific standards, and/or contract requirements, are described in the document's appendices and/or in the project contract. These client requirements are to be performed in addition to the standard processes described in this document.



## Achieving Project Objectives through Standards Compliance

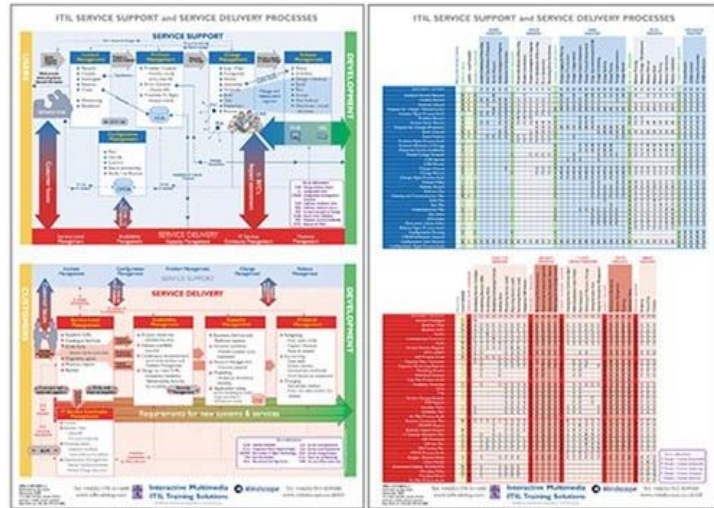
CyberData understands that our success on client contracts relies on sound management and oversight, cost effectiveness, optimal utilization of resources, and end-to-end project quality management.

We recognize the value of standards-based frameworks such as CMMI, ITIL, and ISO, and our incorporation of best practices principles into our management methodology gives our customers a "best of breed" approach to managing IT projects in a way that increases responsibility and effectiveness while improving cost-efficiency and resource usage:

- **The Software Engineering Institute's (SEI) Capability Maturity Model Integration (CMMI)** is a process improvement approach that provides organizations with the essential elements of effective processes that ultimately improve their performance.  
**CMMI-DEV** is used for process improvement in development organizations. CMMI-DEV is a model or collection of "best practices" that organizations follow to dramatically improve the effectiveness, efficiency, and quality of their product and service development work.
- The **Information Technology Infrastructure Library (ITIL)** is a set of concepts and practices for managing Information Technology (IT) services (ITSM), IT development, and IT operations.
- The **International Organization for Standards (ISO) 9000 Quality Standard** provides a number of requirements which an organization needs to fulfill to achieve customer satisfaction through consistent products and services which meet customer

expectations. It includes a requirement for continual (i.e. planned) improvement of the Quality Management System.

- Total Quality Management (TQM)** seeks to integrate all organizational functions (marketing, finance, design, engineering, and production, customer service, etc.) to focus on meeting customer needs and organizational objectives. TQM empowers the Total organization, from the employee to the CEO, with the responsibility of ensuring Quality in their respective products and services, and Management of their processes through the appropriate process improvement channels.



- Lean Six Sigma** is a business improvement methodology that maximizes client value by achieving the fastest rate of improvement in customer satisfaction, cost, quality, process speed, and invested capital.

CyberData’s goals in adopting and integrating CMMI, ITIL, and other standards into our integrated Project Quality Management model (PQM) are to bring to our customers resources and tools that promote:

- Better management of the Total Customer Experience in order to improve customer satisfaction
- Accountability to reduce the risk of not meeting the business requirements
- Process certification to reduce costs and increase communication flow
- Proven business value with scalable solutions
- Continuous improvement through quality assurance processes that repeatedly follow the philosophy of plan – do – check – act

## **Method Activities**

This section describes the activities that make up the CyberData Project Quality Management method. Each activity describes its objective, the project phase in which the activity occurs, the tasks that make up the activity, the deliverables (or work products) that result from the activity, and responsibilities for participants in the activity.

In some cases, key considerations are discussed for the activity. These considerations describe guidance that the Project Manager, and activity participants, should consider in the execution of the activity. Similarly, task guidance may be provided for the tasks in the activity. This guidance suggests an approach or describes factors about the task to be considered during execution of the task.

**PQM 1.1 Define Quality Management Plan**

**Objective**

The objective of this activity is to outline the processes and work products that will be used to verify that the project meets the defined requirements and its deliverables meet client expectations. Quality Management spans several stages of the project, beginning with specific work product reviews, followed by procedure reviews, through functionality testing.



**Phase**

This activity is performed in the Definition phase.

Tasks	
PQM 1.1.1	Schedule meeting with project team leadership
PQM 1.1.2	Meet with project team leadership; identify project scope, quality management objectives, overall project management plan, project schedule, and deliverables
PQM 1.1.3	Schedule meeting with client's quality management point of contact
PQM 1.1.4	Meet with client's quality management point of contact <ul style="list-style-type: none"> <li>• Collect copies of client's documented quality management process and requirements</li> <li>• Discuss client's undocumented quality management expectations, processes, and requirements</li> <li>• Identify specific quality metric standards for work product acceptance</li> </ul>
PQM 1.1.5	Determine project requirements for client's work product acceptance
PQM 1.1.6	Define review process for project deliverables
PQM 1.1.7	Define testing approach for project software deliverables
PQM 1.1.8	Provide project plan (tasks) and schedule guidance to project manager
PQM 1.1.9	Identify and resolve critical quality management impacts on project schedule with the project manager
PQM 1.1.10	Document critical elements identified in this activity
PQM 1.1.11	Submit draft Quality Management Plan deliverable to Project Manager
PQM 1.1.12	Schedule review of draft Quality Management Plan with project team

Tasks	
PQM 1.1.13	Review, update, and deliver the Quality Management Plan deliverable

### Deliverable

The **Quality Management Plan** (also called the **Quality Assurance Surveillance Plan**) describes the overall project quality monitoring process and how the project team will verify that project deliverables comply with CyberData and client quality requirements.

The Plan highlights the project deliverable review and gating requirements described in CyberData’s project execution methodology. The Plan describes the process that the project will engage to review and verify project deliverables, complete the client’s deliverable acceptance requirements, and close-out each phase.

The Plan describes how the project’s work products will be tested, according to a variety of testing stages and types. This description focuses on the testing approach at a high level. This description includes items such as testing tools, which often take time to procure and should be decided early in the project.

The Plan describes the elements of testing that are in scope, organizational elements in scope, as well as the key deliverables required to support testing. To maximize client acceptance of project deliverables, the Plan integrates critical components of the client’s standards, methods, deliverables, and gating criteria (including quality metric standards).

The Plan provides the initial description of Configuration Management and Version Control practices, which may be spun off later into separate project deliverables.

### Responsibilities

- R – Quality Management Specialist
- A – Quality Management Supervisor/Manager, Project Manager
- S – Business Analyst
- C – Project Architect
- I – Technical Team Lead

### Key Considerations

- Specific quality metric standards must be identified and documented at this stage, particularly the standards for accepting a final software build for release to the client’s production environment. The practices and tools engaged by the project to support a “half-percent minor defects or less identified” standard could be significantly different than what is required to support a “zero identified defects” standard.

- The Quality Management Plan establishes the scope of quality review and testing activities throughout the project: which phases, which work products, which categories of review techniques (document reviews, procedure reviews, and/or software reviews)
- Estimating rule-of-thumb – Quality Assurance activities (Unit Test, Integration Test, System Test, User Acceptance Test, and any client Independent Verification & Validation (IV&V)) tend to consume 20-30% of the project schedule, and 50-70% of the project resources during that time.
- The System Test Plan may not deviate from this Project Quality Management method definition without written approval from the program manager and/or the CyberData President.

## Task Guidance

### **Meet with project team leadership; identify project scope, quality management objectives, overall project management plan, project schedule, and deliverables**

This task defines exactly what the Project Quality Management Team needs to test versus what is being tested by another team or is deemed not necessary to test. This task is critical to making sure that the entire project team is in agreement with what the Team will focus on.

Expectations need to be set at the beginning of this activity to make sure there are no surprises in later phases.



### **Meet with client's quality management point of contact**

At the beginning of any project, it is critical that each team understands what needs to be delivered and what criteria will be used to evaluate those deliverables. For testing, understanding the client's quality requirements and quality management process is the base standard for defining quality management approach, reviewing project deliverables, developing test cases, making sure that all requirements have been covered by test cases, and that project deliverables meet the client's quality metric requirements.

### **Define testing approach for project software deliverables**

When analyzing the scope of the project, it is also important at the same time to think about what approach or techniques will be used for each phase of testing. This activity will drive the required skill sets of the team as well as the team size and structure. Here are a few questions that should be considered when defining the testing approach:

- Are there any data security considerations?
- Is there a dedicated test environment or will it be shared?
- Are tests going to be manual or automated?
- What are all the source data systems that are in scope and are there any unique challenges with any of them?
- How are test data sets going to be generated?
- What does the architecture of the system and what are the logical testing points that the team should focus on?

Careful analysis will be required to ensure that the Quality Management Team will be covering all the appropriate areas utilizing the proper testing techniques.

- Test Analysts will have access to Business Analysts
- Requirements will be frozen at integration test cycle - the only changes after the initial build are due to defects. Any requested changes outside of defects will be subject to a change control process in which the Quality Management Specialist will participate
- All necessary code has been delivered to Quality Management Team for “official” builds per iteration
- Once an iteration has been completed, any changes made to that phase will require regression testing
- Consistent and continuing development support throughout the testing cycle
- Unit testing has been successfully completed prior to build submission to the Quality Management Team
- Build information and installation details are accurate and complete upon delivery to the Quality Management Team
- The appropriate test environment is available when the build is delivered
- Each testing activity will have quality metric criteria established, which will be met before code can move on to the next testing activity

### **Provide project plan (tasks) and schedule guidance to project manager**

Time estimates for project quality management activities often start with a “rule of thumb” or analogous estimate, which is progressively refined in each phase of the project.

*In all cases, refinements of the Project Quality Management activities’ time estimates must be reviewed during the phase in which those estimates are revised, and sign-off by the assigned Quality Management Specialist or Quality Management Supervisor is required before the revised estimate may be published and used.*

In the Definition phase, analogous estimates (using actual time from similarly-sized projects) may be used if accurate accounting records of quality management time is readily available to the project team. Alternatively, a “rule of thumb” estimate may be used, such as:

- Phase-level quality management activities will consume 10% of the phase’s schedule and resources
- Testing phase activities and resourcing will consume an estimated 30-50% of the project’s schedule and resources estimate for the Development phase.

- The above estimate could double where the client engages a third-party for Independent Verification and Validation (IV&V) services and/or other business partners that are developing software that will be tested during the project's scheduled activities.
- For the simplest projects, the minimum Testing phase time and documentation/release preparation time is five (5) business days

In the Requirements phase, the Quality Management Specialist will refine the project quality management time estimate(s), based on a bottom-up (task level) analysis of schedule and effort requirements for conducting Project Quality Management activities. The Quality Management Supervisor will guide and assist the Quality Management Specialist in these estimating efforts.

As the project progresses, the Quality Management Specialist will refine these estimates and will provide updates before the internal review of the phase's close-out deliverables.

**Document critical elements identified in this activity**

At the end of this activity, the following items should be clearly defined:

- Testing scope and approach
- Testing schedule
- Testing Team organization (client and project personnel)
- Project deliverable evaluation process and criteria
- Metrics set, including exit criteria from each testing phase
- Project plan, tasks, schedule, and resource estimates for project quality management activities

**PQM 1.9 Execute Definition Phase Deliverable Review**

**Objective**

The objective of this activity is to conduct the processes that review the quality of the phase work products, verifying that the project deliverables meet the defined requirements and client expectations, achieve client acceptance, and close-out the project phase.



**Phase**

This activity is performed in the Definition phase.

Tasks	
PQM 1.9.1	Prepare and Publish Deliverable Statement(s) of Intent (DSOIs) to Client
PQM 1.9.2	Review Phase DSOIs with Client
PQM 1.9.3	Document acceptance requirements for each deliverable
PQM 1.9.4	Document client and internal requirements for phase close-out
PQM 1.9.5	Schedule internal review(s) of phase deliverables
PQM 1.9.6	Schedule client review(s) of phase deliverables
PQM 1.9.7	Prepare checklist of acceptance requirements for each deliverable
PQM 1.9.8	Prepare checklist of phase close-out requirements
PQM 1.9.9	Inventory the phase deliverables
PQM 1.9.10	Conduct internal review(s) of phase deliverables <ul style="list-style-type: none"> <li>Review deliverable(s) against acceptance requirements</li> <li>Identify and document compliance issues, if any</li> <li>Identify and document remediation actions, if needed</li> <li>Schedule follow-up internal review, if needed</li> </ul>
PQM 1.9.11	Conduct client review(s) of phase deliverables <ul style="list-style-type: none"> <li>Review deliverable(s) against acceptance requirements</li> <li>Identify and document compliance issues, if any</li> <li>Identify and document remediation actions, if needed</li> <li>Schedule follow-up client review, if needed</li> <li>Document review results and publish to client</li> </ul>

Tasks	
PQM 1.9.12	Deliver phase deliverables to client for approval and sign-off
PQM 1.9.13	Collect client sign-off on phase deliverables
PQM 1.9.14	Schedule internal review of phase close-out requirements
PQM 1.9.15	Conduct internal review of phase close-out requirements <ul style="list-style-type: none"> <li>• Review status of phase deliverables against close-out requirements</li> <li>• Review status of remaining close-out requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up internal review, if needed</li> <li>• If ready, complete phase close-out</li> </ul>
PQM 1.9.16	Collect and document Lessons Learned.

### Deliverable

A **Deliverable Statement of Intent** is prepared for each deliverable to be produced and approved in the phase. The DSOI describes the planned deliverable in sufficient detail that the client is able to evaluate the planned organization and content of the deliverable, and comment on the suitability of the planned deliverable. Achieving this agreement early in the phase is critical to deliverable acceptance later in the phase.

**Lessons Learned** are documented and collected at the end of each project phase, highlighting activities in need of adjustment or improvement to meet project objectives. Separate sessions are held for the internal project team, and the project team leadership and client participants of the project phase.

### Responsibilities

R – Project Manager

A – Project client

S – All project team members

I – Client contracting officer

### Exit Criteria

Exit Criteria include certain deliverables which must be reviewed and approved (internally and with the client) before the project may proceed to the next phase. In addition to deliverables defined and established in the method(s) that will be engaged by project(s), as well as the

method(s) used by the client and those specified in the project contract, the following deliverables must be completed, reviewed, and approved before this phase is completed:

- **Scope Definition**, including a description of the scope management process and change control board's operation.
- **Configuration Management Plan**, describing the process to recommend and approve changes to software, hardware, and supporting documentation.
- **Work Breakdown Structure**, described at least to level two, which includes phases and major activities. The next phase must be described to the level where specific time estimates are included, team members are assigned, work products are defined, and work time is recorded in the time tracking system. Any activity with a time estimate greater than 40 hours, or a duration greater than two weeks, should be broken down to the next level of detail.
- **Communications Plan**, describing how the project will manage project-specific communications. Where an existing program-level communications plan already exists, this communications plan will describe any supplemental communications that are to be performed at the project level.
- **Deliverable Schedule**, listing all internal and contractual deliverables, for the project, and current due dates.
- **Project Decision Log**, listing all project decisions made, including date of decision and all participants in the decision.
- **Project Risk Log**, listing all identified project risks, including contingency and mitigation plans (and due date and responsible parties for those plans).
- **Project Issue Log**, listing all project issues, response plans, due dates, and responsible parties for each issue.
- **Project Status Reports**, copies of all status reports delivered to the project client during the phase.

### Key Considerations

- All contract deliverables associated with this phase must be signed off by the client for the phase to be considered closed out.
- All internal deliverables associated with this phase must complete the phase review for the phase to be considered closed out.

### Task Guidance

#### **Publish Deliverable Statement(s) of Intent (DSOIs) to Client**

At the start of the project phase, prepare a **Deliverable Statement of Intent** for each deliverable to be produced and approved in the phase. A DSOI is a 1-2 page overview of the deliverable, and will typically define the deliverable format, contents, objectives and goals, reviewers, and acceptance criteria. The contents are often described in *table of contents*

format, with a brief description of each section, which may be expanded or refined in the final deliverable.

**Prepare checklist of acceptance requirements for each deliverable**

The Project Manager, or designated project team member, will review the project contract to identify all processes, tasks, deliverables, and work products that are to be produced during this phase of the project. All identified items are to be included in the checklist of acceptance requirements, for review during this activity.

**PQM 2.1 Prepare System Test Plan**

**Objective**

The objective of this activity is to outline the testing approach to verify that business requirements are met and functioning as designed. Testing spans several stages of the project, beginning with specific functional unit tests that target specific elements of the design, such as single business transactions. As the configuration becomes finalized, the functional testing becomes more integrated and resembles true business scenarios.



**Phase**

This activity is performed in the Requirements phase.

REQUIREMENTS PHASE

Tasks	
PQM 2.1.1	Schedule meeting with project team leadership
PQM 2.1.2	Meet with project team leadership; identify scope of each quality assurance activity; refine testing schedule and resource requirements
PQM 2.1.3	Refine details of each testing activity's scope <ul style="list-style-type: none"> <li>describe the overall types of testing</li> <li>describe the use of automation in testing processes</li> </ul>
PQM 2.1.4	Prepare work breakdown structure (WBS), defining testing activities and tasks for each testing activity
PQM 2.1.5	Define testing team requirements for each testing activity, including business and technical SMEs
PQM 2.1.6	Define artifact requirements for each testing activity, including technology environment requirements, documentation, plans, and test data
PQM 2.1.7	Refine client's quality metric standards into specific metrics for work products that are evaluated by each testing activity

Tasks	
PQM 2.1.8	Define defect management procedure, including <ul style="list-style-type: none"> <li>• Defect identification timing and process</li> <li>• Documenting defects, including prioritization and assignment</li> <li>• Defect tracking and reporting</li> <li>• Defect remediation and repair verification</li> </ul>
PQM 2.1.9	Define automated testing plan, including: <ul style="list-style-type: none"> <li>• Scope of automated testing activities</li> <li>• Identify automated testing tool(s)</li> <li>• Criteria for selection of tests for automation</li> <li>• Test script development procedure</li> <li>• Managing the test script library</li> <li>• Managing builds for regression testing</li> </ul>
PQM 2.1.10	Define performance test plans, including <ul style="list-style-type: none"> <li>• Linking service level agreements (SLAs) to specific transactions</li> <li>• Selecting transactions for performance testing and monitoring</li> <li>• Identify performance test tool(s)</li> </ul>
PQM 2.1.11	Define test plans for system fail-over and disaster recovery
PQM 2.1.12	Define the roles and responsibilities of the Quality Management Team
PQM 2.1.13	Update project plan (tasks) and schedule guidance to project manager
PQM 2.1.14	Identify and resolve critical quality management impacts on project schedule with the project manager
PQM 2.1.15	Submit draft System Test Plan deliverable to Project Manager
PQM 2.1.16	Schedule review of draft System Test Plan with project team
PQM 2.1.17	Review, update, and deliver the System Test Plan deliverable

### Deliverable

The **System Test Plan** outlines the overall project test strategy, describing how the system will be tested according to a variety of testing activities and types. The System Test Plan describes the elements of testing that are in scope as well as the key deliverables required to support testing. This approach includes items such as testing tools, which often take time to procure and should be decided early in the project. The Plan highlights the quality management activities and deliverables of the Testing phase.

The Plan focuses on the testing approach at a high level; it describes the applicable testing activities and the interrelationships between these testing activities.

The Plan highlights the activities of each testing activity, including:

- Scope of the testing in the activity
- Description of each activity
- Resource requirements for conducting testing
- Organization of testing participants and stakeholders, including their responsibilities

The plan describes the processes to be engaged in the testing activities, including defect tracking and reporting, defect remediation and validation, and automated testing.

The Quality Management Specialist defines the common information across the multiple testing activities including items like overall purpose, scope and approach, team organization, schedules, and work breakdown structure (WBS). The WBS describes the organization and sequence of testing tasks by activity.

The System Test Plan describes the processes, team organization, and responsibilities for supporting the client's IV&V activities.

The System Test Plan drives the content and the boundaries across the Integration, System and User Acceptance Test activities. For example:

*Unit Testing falls into the Development team's domain and responsibility. Prior to beginning formal testing procedures, the Development team will provide to the Quality Management team the Release Candidate package.*

In addition, the roles and responsibilities of the project team during the Testing phase are defined, so it is clear what the project team members are accountable for during the Testing phase.

This document is used by the architects, designers, and developers to validate the scope and approach that the testing team is taking for each testing activity.

## Responsibilities

R – Quality Management Specialist

A – Quality Management Supervisor/Manager, Project Manager

S – Business Analyst

C – Project Architect

I – Technical Team Lead

## Key Considerations

- In the Requirements phase, the Quality Management Specialist becomes part of the Analysis Team, participating in client requirements meetings to understand the business and technical context for the new/updated system. The Quality Management Specialist will leverage this direct knowledge when documenting Integration, System, and User

Acceptance test cases and scripts. The Test Team will be responsible for mapping all test activities back to their specified requirements within the Business Requirements Document.

- As the project progresses through the testing phases, testing resources will transition from predominantly CyberData development and testing resources to predominantly client testing resources. Resource requirements should reflect this transition.
- Regression testing is selective retesting of a system or component to verify that modifications and corrections have not caused unintended effects and that the system or component still complies with its specified requirements. Regression testing should be a part of each activity of the testing strategy (Integration, System, and UAT). It should be performed at the end of each testing activity (or at the end of each testing cycle, if multiple cycles are performed within a testing activity) in order to retest and validate any changes, fixes to any defects, or required modifications.
- Schedule delays encountered in Development phase activities, Unit Testing, and/or Unit Tested Release will not be justification for changing the System Test Plan, nor its scope, activities, schedule, or effort commitments. Once Development phase activities commence, any proposed changes to the Testing phase schedule must be approved by the Program Manager and/or the CyberData President before they are published and used.

REQUIREMENTS PHASE

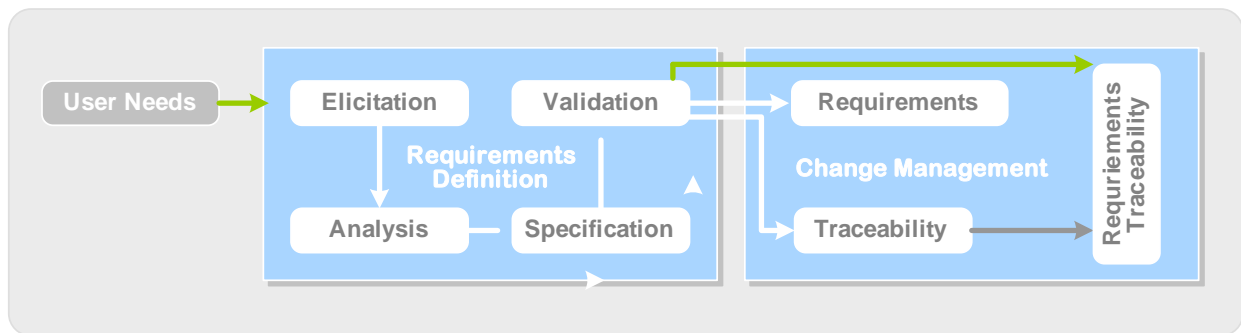


Figure 5 – CyberData Requirements Engineering Process

### Task Guidance

#### Refine details of each testing activity’s scope

In the Execute Integration Testing activity, the Quality Management Specialist should outline in detail how the design (e.g. output of the data warehousing and business intelligence processes) will be verified. Integration tests are specified to test interaction between software components, hardware components, or both. Test plans are designed to evaluate the interaction between all areas of the system interface to ensure that components interact with each other correctly and that there are no gaps in the data flow. The final Integration Test should prove that system works as an integrated unit when all the fixes are complete. The

breakdown of the functionality to be tested in the integration cycle will be specified in the approach in the detailed test plan.

The Execute System Testing activity is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. The breakdown of the functionality to be tested in this activity will be specified in the approach for this activity. The Plan should summarize how the system requirements (output of the end-user reporting tool) will be validated.

User Acceptance Testing is planned and executed by the Business Representative(s). Completion of UAT ensures that the system operates in the manner expected, and any supporting material such as procedures, forms etc. are accurate and suitable for the purpose intended. It is high level testing, ensuring that there are no gaps in functionality.

### **Refine client's quality metric standards into specific metrics for work products that are evaluated by each testing activity**

In the Definition phase, the client identifies their quality metric requirements, which describe the minimum quality metrics for an acceptable delivery of the project's deliverable(s). The System Test Plan should describe the minimal metrics for each testing activity, with the final activity delivering to the client's quality metric requirements.

### **Update project plan (tasks) and schedule guidance to project manager**

Time estimates for project quality management activities often start with a "rule of thumb" or analogous estimate, which is progressively refined in each phase of the project.

*In all cases, refinements of the Project Quality Management activities' time estimates must be reviewed during the phase in which those estimates are revised, and sign-off by the assigned Quality Management Specialist or Quality Management Supervisor is required before the revised estimate may be published and used.*

In the Requirements phase, the Quality Management Specialist will refine the project quality management time estimate(s) provided in the Definition phase, based on a bottom-up (task level) analysis of schedule and effort requirements for conducting Project Quality Management activities.

Estimates for the Testing phase activities will be influenced by:

- the number of requirements in the Requirements deliverable(s),
- the number of test cases required,
- the number of iterations of Integration, System, and User Acceptance testing
- the client's specific testing process requirements, including documentation preparation

The Quality Management Supervisor will guide and assist the Quality Management Specialist in these estimating efforts.

Another important consideration in producing accurate, reliable estimates is the project team's Scope Management and Requirements Gathering process(es). Scope changes that are managed through a Change Control Board (CCB) often include impact assessments, particularly to the processes engaged, project schedule, and resource effort estimates. In these

cases, estimates for project quality management activities are updated to account for the scope change. Similarly, significant changes in Requirements are managed by this process.

Failure to properly manage these processes could result in delays to all phases of the project. Allowing Requirements to continue to be collected into the Design and Development phases will almost guarantee that those phases will miss their schedules. Delays in Design and Development phases will not be justification for changing the System Test Plan, nor its scope, activities, schedule, or effort commitments.

As the project progresses, the Quality Management Specialist will refine these estimates and will provide updates before the internal review of the phase's close-out deliverables.

**PQM 2.9 Execute Requirements Phase Deliverable Review**

**Objective**

The objective of this activity is to conduct the processes that review the quality of the phase work products, verifying that the project deliverables meet the defined requirements and client expectations, achieve client acceptance, and close-out the project phase.



**Phase**

This activity is performed in the Requirements phase.

REQUIREMENTS PHASE

Tasks	
PQM 2.9.1	Prepare and Publish Deliverable Statement(s) of Intent (DSOIs) to Client
PQM 2.9.2	Review Phase DSOIs with Client
PQM 2.9.3	Document acceptance requirements for each deliverable
PQM 2.9.4	Document client and internal requirements for phase close-out
PQM 2.9.5	Schedule internal review(s) of phase deliverables
PQM 2.9.6	Schedule client review(s) of phase deliverables
PQM 2.9.7	Prepare checklist of acceptance requirements for each deliverable
PQM 2.9.8	Prepare checklist of phase close-out requirements
PQM 2.9.9	Inventory the phase deliverables
PQM 2.9.10	Conduct internal review(s) of phase deliverables <ul style="list-style-type: none"> <li>• Review deliverable(s) against acceptance requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up internal review, if needed</li> </ul>
PQM 2.9.11	Conduct client review(s) of phase deliverables <ul style="list-style-type: none"> <li>• Review deliverable(s) against acceptance requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up client review, if needed</li> <li>• Document review results and publish to client</li> </ul>

Tasks	
PQM 2.9.12	Deliver phase deliverables to client for approval and sign-off
PQM 2.9.13	Collect client sign-off on phase deliverables
PQM 2.9.14	Schedule internal review of phase close-out requirements
PQM 2.9.15	Conduct internal review of phase close-out requirements <ul style="list-style-type: none"> <li>• Review status of phase deliverables against close-out requirements</li> <li>• Review status of remaining close-out requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up internal review, if needed</li> <li>• If ready, complete phase close-out</li> </ul>
PQM 2.9.16	Collect and document Lessons Learned

### Deliverable

A **Deliverable Statement of Intent** is prepared for each deliverable to be produced and approved in the phase. The DSOI describes the planned deliverable in sufficient detail that the client is able to evaluate the planned organization and content of the deliverable, and comment on the suitability of the planned deliverable. Achieving this agreement early in the phase is critical to deliverable acceptance later in the phase.

**Lessons Learned** are documented and collected at the end of each project phase, highlighting activities in need of adjustment or improvement to meet project objectives. Separate sessions are held for the internal project team, and the project team leadership and client participants of the project phase.

### Responsibilities

R – Project Manager

A – Project client

S – All project team members

I – Client contracting officer

### Exit Criteria

Exit Criteria include certain deliverables which must be reviewed and approved (internally and with the client) before the project may proceed to the next phase. In addition to deliverables defined and established in the method(s) that will be engaged by project(s), as well as the method(s) used by the client and those specified in the project contract, the following deliverables must be completed, reviewed, and approved before this phase is completed:

- **Business Requirements**, including detailed functional and non-functional requirements for the system to be delivered by the project. This deliverable must include data requirements and business rules.
- **Requirements Traceability Matrix (RTM)**, listing all requirements in requirements ID sequence. The RTM should indicate the source of the requirement and the date collected.
- **Work Breakdown Structure (updated)**, all remaining phases must be described to the level where specific time estimates are included, team members are assigned, work products are defined, and work time is recorded in the time tracking system. Any activity with a time estimate greater than 40 hours, or a duration greater than two weeks, should be broken down to the next level of detail.
- **Deliverable Schedule (updated)**, listing all internal and contractual deliverables, for the project, and current due dates.
- **Project Decision Log (updated)**, listing all project decisions made, including date of decision and all participants in the decision.
- **Project Risk Log (updated)**, listing all identified project risks, including contingency and mitigation plans (and due date and responsible parties for those plans).
- **Project Issue Log (updated)**, listing all project issues, response plans, due dates, and responsible parties for each issue.
- **Project Status Reports**, copies of all status reports delivered to the project client during the phase.

### Key Considerations

- All contract deliverables associated with this phase must be signed off by the client for the phase to be considered closed out.
- All internal deliverables associated with this phase must complete the phase review for the phase to be considered closed out.

### Task Guidance

#### **Publish Deliverable Statement(s) of Intent (DSOIs) to Client**

At the start of the project phase, prepare a **Deliverable Statement of Intent** for each deliverable to be produced and approved in the phase. A DSOI is a 1-2 page overview of the deliverable, and will typically define the deliverable format, contents, objectives and goals, reviewers, and acceptance criteria. The contents are often described in *table of contents* format, with a brief description of each section, which may be expanded or refined in the final deliverable.

#### **Prepare checklist of acceptance requirements for each deliverable**

The Project Manager, or designated project team member, will review the project contract to identify all processes, tasks, deliverables, and work products that are to be produced during this

phase of the project. All identified items are to be included in the checklist of acceptance requirements, for review during this activity.

**PQM 3.1 Prepare Quality Architecture Design**

**Objective**

The objective of this activity is to establish the approach for continuous quality management in the delivered system. The architecture leverages the quality functional requirements, as described in the project’s Requirements deliverable, to guide its scoping and design activities. The Quality Management Specialist collaborates with the Project Architect to verify that the system architecture addresses the project’s quality requirements.



**Phase**

This activity is performed in the High-Level Design/Architecture Design phase.

Tasks	
PQM 3.1.1	Review Requirements deliverable
PQM 3.1.2	Excerpt individual quality-related requirements
PQM 3.1.3	Participate in project architecture analysis and design activities, focusing on project quality requirements
PQM 3.1.4	Map architecture elements to quality-related requirements; verify adequate coverage of all quality-related requirements
PQM 3.1.5	As architecture design artifacts emerge, excerpt quality architecture elements
PQM 3.1.6	Assemble quality architecture elements into Quality Architecture Design document
PQM 3.1.7	Review Quality Architecture Design with Project Architect
PQM 3.1.8	Update project plan (tasks) and schedule guidance to project manager
PQM 3.1.9	Identify and resolve critical quality management impacts on project schedule with the project manager
PQM 3.1.10	Submit draft Quality Architecture Design deliverable to Project Manager
PQM 3.1.11	Schedule review of draft Quality Architecture Design with project team
PQM 3.1.12	Review, update, and deliver the Quality Architecture Design deliverable

HIGH-LEVEL DESIGN PHASE

## Deliverable

The **Quality Architecture Design** documents the techniques and technologies to be used to manage each category of quality management throughout the system to be delivered. This deliverable excerpts and highlights aspects of the overall system architecture that address quality management.

## Responsibilities

- R – Quality Management Specialist
- A – Quality Management Supervisor/Manager, Project Manager
- S – Business Analyst, Project Architect
- I – Technical Team Lead

## Key Considerations

- In most cases, quality management processes should operate independently of the transactions that they are verifying. However, there will be cases when specific verification actions will be integrated with the transactions themselves. Both sets of processes should be described in the Quality Architecture Design

## Task Guidance

### **Update project plan (tasks) and schedule guidance to project manager**

Time estimates for project quality management activities often start with a “rule of thumb” or analogous estimate, which is progressively refined in each phase of the project.

*In all cases, refinements of the Project Quality Management activities’ time estimates must be reviewed during the phase in which those estimates are revised, and sign-off by the Project Manager, and the assigned Quality Management Specialist or Quality Management Supervisor, is required before the revised estimate may be published and used.*

In the Design phase, the Quality Management Specialist will refine the project quality management time estimate(s) provided in the Requirements phase, based on a bottom-up (task level) analysis of schedule and effort requirements for conducting Project Quality Management activities.

By the Design phase, most refinements to the schedule and effort estimates should be based on new information and/or clarifications in project scope, user requirements, development activities, and/or testing activities.

The Quality Management Supervisor will guide and assist the Quality Management Specialist in these estimating efforts.

As the project progresses, the Quality Management Specialist will refine these estimates and will provide updates before the internal review of the phase’s close-out deliverables.

**PQM 3.2 Prepare Testing Environment Design**

**Objective**

The objective of this activity is to define and design the technology infrastructure required to support testing activities later in the project. The Quality Management Specialist collaborates with the Project Architect to verify that the testing environment design addresses the project’s testing requirements.



**Phase**

This activity is performed in the High-Level Design/Architecture Design phase.

Tasks	
PQM 3.2.1	Review Requirements deliverable and Service Level Agreement(s) (SLAs)
PQM 3.2.2	Excerpt individual technology- and transaction volume-related requirements
PQM 3.2.3	Participate in project architecture analysis and design activities, collecting and reviewing technology component specifications
PQM 3.2.4	As the Technology Architecture Design artifacts emerge, review inventory of technology elements that will be used to support Production processing
PQM 3.2.5	Evaluate Production SLAs against testing requirements
PQM 3.2.6	Adjust capacity specifications, as needed to support testing requirements
PQM 3.2.7	Assemble technology elements into Testing Environment Design document
PQM 3.2.8	Review Testing Environment Design with Project Architect
PQM 3.2.9	Submit draft Testing Environment Design deliverable to Project Manager
PQM 3.2.10	Schedule review of draft Testing Environment Design with project team
PQM 3.2.11	Review, update, and deliver the Testing Environment Design deliverable

HIGH-LEVEL DESIGN PHASE

**Deliverable**

The **Testing Environment Design** documents the technologies, and their capacities, to be used to execute the testing activities for the project (Unit, Integration, System, User Acceptance,

and Performance). The Testing Environment should leverage much of the overall Technical Architecture Design, particularly focusing on the Production infrastructure design.

## Responsibilities

R – Quality Management Specialist

A – Quality Management Supervisor/Manager, Project Manager

S – Business Analyst, Project Architect

I – Technical Team Lead

## Key Considerations

- The Testing Environment may be required to support more than one type of testing environment:
  - a Unit Testing environment, in which remediated software components are introduced at will by the developers;
  - a Solution Integration Testing environment, in which representatives of two or more project teams test the integrated execution of their respective systems;
  - a Formal Testing environment, in which software components are introduced by build and managed by the Quality Management Specialist(s); and
  - a User Acceptance Testing environment, in which representatives of the user community will select and execute a variety of tests.
- The Testing Environment must be designed to operate independently of the Development Environment or other environments. No shared components are to be permitted, including hardware servers, data storage, software servers, databases, catalogues, and others. A shared network may be permitted, when independent networks would not be practical.
- The Testing Environment Design should designate which servers will be put to which purpose (such as database server, business intelligence server, webserver, etc.). The design should specify which server(s) and/or testing environment(s) will be used exclusively by Quality Management Specialists and which will be used by the rest of the project team.

**PQM 3.9 Execute High-Level Design Phase Deliverable Review**

**Objective**

The objective of this activity is to conduct the processes that review the quality of the phase work products, verifying that the project deliverables meet the defined requirements and client expectations, achieve client acceptance, and close-out the project phase.



**Phase**

This activity is performed in the High-Level Design/Architecture Design phase.

Tasks	
PQM 3.9.1	Prepare and Publish Deliverable Statement(s) of Intent (DSOIs) to Client
PQM 3.9.2	Review Phase DSOIs with Client
PQM 3.9.3	Document acceptance requirements for each deliverable
PQM 3.9.4	Document client and internal requirements for phase close-out
PQM 3.9.5	Schedule internal review(s) of phase deliverables
PQM 3.9.6	Schedule client review(s) of phase deliverables
PQM 3.9.7	Prepare checklist of acceptance requirements for each deliverable
PQM 3.9.8	Prepare checklist of phase close-out requirements
PQM 3.9.9	Inventory the phase deliverables
PQM 3.9.10	Conduct internal review(s) of phase deliverables <ul style="list-style-type: none"> <li>• Review deliverable(s) against acceptance requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up internal review, if needed</li> </ul>

Tasks	
PQM 3.9.11	Conduct client review(s) of phase deliverables <ul style="list-style-type: none"> <li>• Review deliverable(s) against acceptance requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up client review, if needed</li> <li>• Document review results and publish to client</li> </ul>
PQM 3.9.12	Deliver phase deliverables to client for approval and sign-off
PQM 3.9.13	Collect client sign-off on phase deliverables
PQM 3.9.14	Schedule internal review of phase close-out requirements
PQM 3.9.15	Conduct internal review of phase close-out requirements <ul style="list-style-type: none"> <li>• Review status of phase deliverables against close-out requirements</li> <li>• Review status of remaining close-out requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up internal review, if needed</li> <li>• If ready, complete phase close-out</li> </ul>
PQM 3.9.16	Collect and document Lessons Learned

### Deliverable

A **Deliverable Statement of Intent** is prepared for each deliverable to be produced and approved in the phase. The DSOI describes the planned deliverable in sufficient detail that the client is able to evaluate the planned organization and content of the deliverable, and comment on the suitability of the planned deliverable. Achieving this agreement early in the phase is critical to deliverable acceptance later in the phase.

**Lessons Learned** are documented and collected at the end of each project phase, highlighting activities in need of adjustment or improvement to meet project objectives. Separate sessions are held for the internal project team, and the project team leadership and client participants of the project phase.

### Responsibilities

R – Project Manager

A – Project client

S – All project team members

I – Client contracting officer

## Exit Criteria

Exit Criteria include certain deliverables which must be reviewed and approved (internally and with the client) before the project may proceed to the next phase. In addition to deliverables defined and established in the method(s) that will be engaged by project(s), as well as the method(s) used by the client and those specified in the project contract, the following deliverables must be completed, reviewed, and approved before this phase is completed:

- **Functional Specifications**, describing how the system that will be built by the project will meet the business requirements.
- **Requirements Traceability Matrix (updated)**, listing all requirements in requirements ID sequence. The RTM should indicate the Functional Specifications ID(s) that address a requirement.
- **Data Models**, including subject area diagram, conceptual models, and logical data model. These are requirements-oriented models. The Physical data model is a design-oriented model that is prepared in the next phase.
- **Data Profiles**, analysis results of profiling candidate source systems, and recommendations for designations of authoritative sources of enterprise data.
- **Data Dictionary**, describing the business definition of all in-scope data elements to be used by the project.
- **ETL Data Transformation Rules**, describing the data source (database, table(s), column(s)), transformation description, cleansing rules, data rejection rules, and data target (database, table(s), column(s))
- **Architecture Design**, including business, data, applications, security, and technology architecture designs and models.
- **Work Breakdown Structure (updated)**, all remaining phases must be described to the level where specific time estimates are included, team members are assigned, work products are defined, and work time is recorded in the time tracking system. Any activity with a time estimate greater than 40 hours, or a duration greater than two weeks, should be broken down to the next level of detail.
- **Deliverable Schedule (updated)**, listing all internal and contractual deliverables, for the project, and current due dates.
- **Project Decision Log (updated)**, listing all project decisions made, including date of decision and all participants in the decision.
- **Project Risk Log (updated)**, listing all identified project risks, including contingency and mitigation plans (and due date and responsible parties for those plans).
- **Project Issue Log (updated)**, listing all project issues, response plans, due dates, and responsible parties for each issue.
- **Project Status Reports**, copies of all status reports delivered to the project client during the phase.

## Key Considerations

- All contract deliverables associated with this phase must be signed off by the client for the phase to be considered closed out.
- All internal deliverables associated with this phase must complete the phase review for the phase to be considered closed out.

## Task Guidance

### **Publish Deliverable Statement(s) of Intent (DSOIs) to Client**

At the start of the project phase, prepare a **Deliverable Statement of Intent** for each deliverable to be produced and approved in the phase. A DSOI is a 1-2 page overview of the deliverable, and will typically define the deliverable format, contents, objectives and goals, reviewers, and acceptance criteria. The contents are often described in *table of contents* format, with a brief description of each section, which may be expanded or refined in the final deliverable.

### **Prepare checklist of acceptance requirements for each deliverable**

The Project Manager, or designated project team member, will review the project contract to identify all processes, tasks, deliverables, and work products that are to be produced during this phase of the project. All identified items are to be included in the checklist of acceptance requirements, for review during this activity.

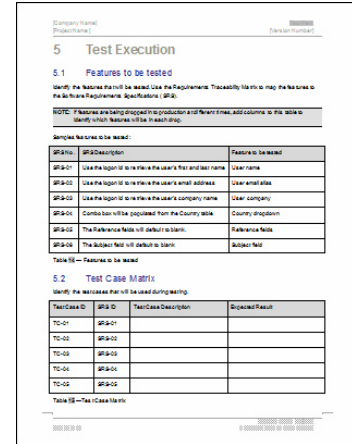
**PQM 4.1 Define Functional Unit Test Cases**

**Objective**

The objective of this activity is to establish the approach for functional unit testing. Functional unit testing concentrates on thoroughly testing individual components, such as individual transactions and development objects. All scenario types, including positive and negative variations, should be included in this approach. Manual and automated testing processes should be described.

**Phase**

This activity is performed in the Low-Level/Application Program Design phase.



Tasks	
PQM 4.1.1	Review Requirements and Architecture Design deliverable
PQM 4.1.2	For each requirement, identify testing technique(s) needed to verify compliance
PQM 4.1.3	Map tests to application programs <ul style="list-style-type: none"> <li>Using the Requirements Traceability Matrix (RTM), map requirements to application programs</li> <li>Map testing techniques to each application program</li> <li>Update Requirements Traceability Matrix, mapping test cases to requirements</li> </ul>
PQM 4.1.4	Assemble test cases -- list of requirements to be tested, and tests to be performed, for an application program
PQM 4.1.5	Annotate test cases to indicate which tests will be automated, and which will be performed manually
PQM 4.1.6	Assemble test cases into Functional Unit Test Cases document
PQM 4.1.7	Review and communicate Unit Test quality metrics, established in the System Test Plan, with the Project Team
PQM 4.1.8	Copy the Unit Test quality metrics to the Functional Unit Test Cases deliverable
PQM 4.1.9	Submit draft Functional Unit Test Cases deliverable to Project Manager
PQM 4.1.10	Schedule review of draft Functional Unit Test Cases with project team
PQM 4.1.11	Review, update, and deliver the Functional Unit Test Cases deliverable

## Deliverable

The **Functional Unit Test Cases** is a documented series of test cases, listing the types of tests to be performed against each application program. Functional unit testing focuses on configuration-related items, such as transactions and program development. The development teams usually conduct functional unit testing.

A Requirements Traceability Matrix (RTM) is created by associating requirements with the solution components that satisfy them. Traceability ensures completeness, that all lower level requirements come from higher level requirements, and that all higher level requirements are allocated to lower level requirements. Traceability is also used to manage change and provides the basis for test planning. To ensure each requirement is tested thoroughly, the Quality Management Specialist associates all test cases with documented requirements. The **updated Requirements Traceability Matrix** (RTM) lists the functional unit test cases that are mapped to specific functional and technical requirements.

## Responsibilities

- R – Quality Management Specialist
- A – Quality Management Supervisor/Manager, Project Manager
- S – Business Analyst
- C – Application Designers/Developers
- I – Technical Team Lead

## Key Considerations

- Although it is being developed concurrently, it's important to maintain awareness of the contents of the Application Design deliverable as it progresses through its drafts.

**PQM 4.2 Define Integration Test Cases**

**Objective**

The objective of this activity is to establish the approach for functional integration testing. Functional integration testing concentrates on thoroughly testing how individual components interact with each other, verifying that interactions between components perform to requirements. Each “application program set” is a pair of components with one or more integration requirements. All scenario types, including positive and negative variations, should be included in this approach. Manual and automated testing processes should be described.



**Phase**

This activity is performed in the Low-Level/Application Program Design phase.

Tasks	
PQM 4.2.1	Review Requirements and Architecture Design deliverable
PQM 4.2.2	For each integration requirement, identify testing technique(s) needed to verify compliance
PQM 4.2.3	Map tests to sets of application programs <ul style="list-style-type: none"> <li>Using the Requirements Traceability Matrix (RTM), map requirements to application program sets</li> <li>Map testing techniques to each application program set</li> <li>Update Requirements Traceability Matrix, mapping test cases to requirements</li> </ul>
PQM 4.2.4	Assemble test cases -- list of requirements to be tested, and tests to be performed, for an application program set
PQM 4.2.5	Annotate test cases to indicate which tests will be automated, and which will be performed manually
PQM 4.2.6	Assemble test cases into Integration Test Cases document
PQM 4.2.7	Review and communicate Integration Test quality metrics, established in the System Test Plan, with the Project Team
PQM 4.2.8	Copy the Integration Test quality metrics to the Integration Test Cases

Tasks	
	deliverable
PQM 4.2.9	Submit draft Integration Test Cases deliverable to Project Manager
PQM 4.2.10	Schedule review of draft Integration Test Cases with project team
PQM 4.2.11	Review, update, and deliver the Integration Test Cases deliverable

### Deliverable

The **Integration Test Cases** deliverable is a documented series of test cases, listing the types of tests to be performed against each interacting application program set. Integration testing focuses on interactions between application components that participate in a transaction.

The deliverable identifies the internal and external components of the system that interconnect and need to be validated in an integrated manner. The deliverable allows the user to:

- Understand the role of every dependency in the integration test
- Understand all inputs to the system integration test
- Understand all outputs of the system integration test

The updated **Requirements Traceability Matrix (RTM)** lists the integration test cases that are mapped to specific functional and technical requirements.

### Responsibilities

- R – Quality Management Specialist
- A – Quality Management Supervisor/Manager, Project Manager
- S – Business Analyst
- C – Application Designers/Developers
- I – Technical Team Lead

### Key Considerations

- Although it is being developed concurrently, it's important to maintain awareness of the contents of the Application Design deliverable as it progresses through its drafts.

### Task Guidance

**For each integration requirement, identify testing technique(s) needed to verify compliance**

Types of integration testing may include, but are not limited to:

- **Source to staging verification** – This type of testing will verify the source system data is properly loaded into the staging environment. The verification is done by performing row counts, evaluation of exception files, and verification of proper data mapping to the target.
- **Staging to history data store load** – This type of testing will verify the ability of the system to properly process (filter, cleanse, load, etc.) loads of data into the history store area of the staging environment as well as ensuring that all required transformations have been performed accurately.
- **Staging to EDW data loading** – This testing will verify the transition of data from the staging environment to the EDW environment. This testing will ensure that row counts in the Staging environment match those in the EDW environment's relational historical repository.
- **EDW to Presentation Layer verification** – This testing will verify the data that is extracted from the relational historical repository of the EDW to the dimensional structures of the EDW. This testing will validate both the structure and the content of the output data. This testing should be applied to both the EDW dimensional structures and the cubes that are built from those structures.

**PQM 4.3 Define System Test Cases**

**Objective**

The objective of this activity is to establish the approach for end-to-end testing. System tests are developed to validate the system as a whole from end-to-end, combining hardware and software application components. System testing concentrates on thoroughly testing from the point at which data enters the system, to each of the end points that use that data, verifying that all interactions between components, from end-to-end, perform to the documented requirements.

All scenario types, including positive and negative variations, should be included in this approach. Some limited negative test cases for processing, missing keys, and FTP transfers are performed as part of a system test case. Manual and automated testing processes should be described.



**Phase**

This activity is performed in the Low-Level/Application Program Design phase.

Tasks	
PQM 4.3.1	Review Requirements and Architecture Design deliverable
PQM 4.3.2	For each type of transaction that can be performed through the system, identify testing technique(s) needed to verify compliance
PQM 4.3.3	Map transaction tests to beginning and end points of the application <ul style="list-style-type: none"> <li>Using the Requirements Traceability Matrix (RTM), map functional transaction requirements to application program end-point sets</li> <li>Map testing techniques to each transaction</li> <li>Update Requirements Traceability Matrix, mapping test cases to requirements</li> </ul>
PQM 4.3.4	Assemble test cases -- list of requirements to be tested, and tests to be performed, for an end-to-end transaction
PQM 4.3.5	Annotate test cases to indicate which tests will be automated, and which will be performed manually
PQM 4.3.6	Assemble test cases into System Test Cases document

Tasks	
PQM 4.3.7	Review and communicate System Test quality metrics, established in the System Test Plan, with the Project Team
PQM 4.3.8	Copy the System Test quality metrics to the System Test Cases deliverable
PQM 4.3.9	Update project plan (tasks) and schedule guidance to project manager
PQM 4.3.10	Identify and resolve critical quality management impacts on project schedule with the project manager
PQM 4.3.11	Submit draft System Test Cases deliverable to Project Manager
PQM 4.3.12	Schedule review of draft System Test Cases with project team
PQM 4.3.13	Review, update, and deliver the System Test Cases deliverable

### Deliverable

The **System Test Cases** deliverable is a documented series of test cases, listing the types of tests to be performed for a functional transaction, against the application system. System testing focuses on the chain of interactions between application end-points that deliver a functional transaction.

The updated **Requirements Traceability Matrix (RTM)** lists the system test cases that are mapped to specific functional and technical requirements.

### Responsibilities

- R – Quality Management Specialist
- A – Quality Management Supervisor/Manager, Project Manager
- S – Business Analyst
- C – Application Designers/Developers
- I – Technical Team Lead

### Key Considerations

- Although it is being developed concurrently, it's important to maintain awareness of the contents of the Application Design deliverable as it progresses through its drafts.

### Task Guidance

#### Update project plan (tasks) and schedule guidance to project manager

Time estimates for project quality management activities often start with a “rule of thumb” or analogous estimate, which is progressively refined in each phase of the project.

*In all cases, refinements of the Project Quality Management activities' time estimates must be reviewed during the phase in which those estimates are revised, and sign-off by the Project Manager, and the assigned Quality Management Specialist or Quality Management Supervisor, is required before the revised estimate may be published and used.*

By the Development phase, the Quality Management Specialist should have sufficient clarity to the project scope, requirements, and construction processes that any refinements are likely to be minor adjustments. Significant changes in time or schedule estimates will likely only result from changes in project scope, which must be managed through the project's Scope Management process.

**PQM 4.4 Define User Acceptance Test (UAT) Cases**

**Objective**

The objective of this activity is to establish the approach for user acceptance testing (UAT). User acceptance testing establishes a set of tests that exercise the application system’s functionality, while establishing credibility with the participating users that the system delivers the required functionality. All scenario types, including positive and negative variations, should be included in this approach. Manual and automated testing processes should be described.



**Phase**

This activity is performed in the Low-Level/Application Program Design phase.

Tasks	
PQM 4.4.1	Review Requirements and Architecture Design deliverable
PQM 4.4.2	Identify the client users that will be participating in UAT
PQM 4.4.3	Identify users’ UAT roles <ul style="list-style-type: none"> <li>Those roles may be one or more of: planning, test case development, test execution, or testing results reporting/review</li> </ul>
PQM 4.4.4	Schedule UAT planning meeting
PQM 4.4.5	Conduct UAT planning meeting <ul style="list-style-type: none"> <li>Highlight planned testing processes (unit, integration, system, and UAT)</li> <li>Discuss and schedule UAT activities</li> <li>Update Requirements Traceability Matrix, mapping test cases to requirements</li> </ul>
PQM 4.4.6	Identify existing test cases that will be conducted during UAT
PQM 4.4.7	Identify new test cases that will be added to UAT
PQM 4.4.8	Identify test data that will be required to support UAT test cases
PQM 4.4.9	Annotate test cases to indicate which tests will be automated, and which will be performed manually
PQM 4.4.10	Assemble test cases into User Acceptance Test Cases document

Tasks	
PQM 4.4.11	Review and communicate User Acceptance Test quality metrics, established in the System Test Plan, with the Project Team
PQM 4.4.12	Copy the User Acceptance Test quality metrics to the User Acceptance Test Cases deliverable
PQM 4.4.13	Submit draft User Acceptance Test Cases deliverable to Project Manager
PQM 4.4.14	Schedule review of draft User Acceptance Test Cases with project team
PQM 4.4.15	Review, update, and deliver the User Acceptance Test Cases deliverable

### Deliverable

The **User Acceptance Test Cases** deliverable is a documented series of test cases, listing the types of tests to be performed by user community representatives, against the application system.

The updated **Requirements Traceability Matrix (RTM)** lists the user acceptance test cases that are mapped to specific functional and technical requirements.

### Responsibilities

- R – Quality Management Specialist
- A – Quality Management Supervisor/Manager, Project Manager
- S – Business Analyst, User Community Representatives
- C – Application Designers/Developers
- I – Technical Team Lead

### Key Considerations

- Although it is being developed concurrently, it's important to maintain awareness of the contents of the Application Design deliverable as it progresses through its drafts.
- The client may wish to conduct multiple User Acceptance Test stages, each with a different or broader audience. Test cases may vary from audience to audience, depending on the focus and priorities of each audience.

**PQM 4.5 Define Performance Test Cases**

**Objective**

The objective of this activity is to establish the approach for performance testing. Performance testing establishes a set of tests that exercise the application system’s technical resources, verifying compliance with established service level agreements (SLAs). Manual and automated process testing should be described.

**Phase**

This activity is performed in the Low-Level/Application Program Design phase.

Tasks	
PQM 4.5.1	Review Requirements and Architecture Design deliverables
PQM 4.5.2	Review service level agreements (SLAs)
PQM 4.5.3	Identify existing test cases that will be conducted during performance testing
PQM 4.5.4	Update Requirements Traceability Matrix, mapping test cases to requirements
PQM 4.5.5	Identify test data that will be required to support performance test cases
PQM 4.5.6	Assemble test cases into Performance Test Cases document
PQM 4.5.7	Review and communicate Performance Test quality metrics, established in the System Test Plan, with the Project Team
PQM 4.5.8	Copy the Performance Test quality metrics to the Performance Test Cases deliverable
PQM 4.5.9	Submit draft Performance Test Cases deliverable to Project Manager
PQM 4.5.10	Schedule review of draft Performance Test Cases with project team
PQM 4.5.11	Review, update, and deliver the Performance Test Cases deliverable

**Deliverable**

The **Performance Test Cases** deliverable is a documented series of test cases, listing the types of performance tests to be performed against the application system.

The updated **Requirements Traceability Matrix** (RTM) lists the performance test cases that are mapped to specific performance requirements. Service level agreement (SLA) metrics may be added to the RTM to make up for the absence of performance requirements that may be in the SLA, but not in the project’s Requirements document.

## Responsibilities

- R – Quality Management Specialist
- A – Quality Management Supervisor/Manager, Project Manager
- S – Business Analyst, User Community Representatives
- C – Application Designers/Developers
- I – Technical Team Lead

## Key Considerations

- Although it is being developed concurrently, it's important to maintain awareness of the contents of the Application Design deliverable as it progresses through its drafts.

**PQM 4.6 Install and Configure Testing Environment**

**Objective**

The objective of this activity is to prepare the non-production technology environment that will be used to conduct testing activities.



**Phase**

This activity is performed in the Low-Level/Application Program Design phase.

Tasks	
PQM 4.6.1	Verify inventory of hardware and software components
PQM 4.6.2	Verify software release and patch versions
PQM 4.6.3	Install and configure the non-production hardware and software
PQM 4.6.4	Verify compliance with prerequisites established for the production environment
PQM 4.6.5	Secure the testing environment from non-authorized access
PQM 4.6.6	Establish non-production network connectivity
PQM 4.6.7	Install, configure, and verify operation of automated testing tools, testing management tools, and defect management tools
PQM 4.6.8	Develop non-production backup and recovery test materials
PQM 4.6.9	Test the non-production backup and recovery test materials
PQM 4.6.10	Document and publish the management procedures for administering the testing environment
PQM 4.6.11	Communicate certification of the Testing Environment to the Project Manager

**Deliverable**

The Quality Management Specialist will communicate the completion of the activity to the Project Manager. The deliverable is the certified **Testing Environment**.

The **Testing Environment Management Procedures** will describe the procedures that the project will engage in managing and administering the testing environment. These procedures will describe:

- Configuration management activities, including approval process(es) for changing hardware or software components.
- Application component management activities, including approval process(es) for introducing updated software components (newly built or remediated components)

### Responsibilities

R – Quality Management Specialist

A – Quality Management Supervisor/Manager, Project Manager

S – Project Technology Architect

C – Technology Infrastructure Manager

I – Technical Team Lead

### Key Considerations

- It is critical that the Testing Environment is synchronized with the Production Environment, either the Production Environment's current state or the state required before the application system will be installed in Production.
- Non-Production Software includes the following software components:
  - Operating System
  - Software
  - Applications
  - Software
  - Packages
  - Tools and Utilities
- Non-Production Network Connectivity includes implementing the network infrastructure, installing the network software, and testing the network connection for the non-production environment.

**PQM 4.9 Execute Low-Level Design Phase Deliverable Review**

**Objective**

The objective of this activity is to conduct the processes that review the quality of the phase work products, verifying that the project deliverables meet the defined requirements and client expectations, achieve client acceptance, and close-out the project phase.



**Phase**

This activity is performed in the Low-Level/Application Program Design phase.

Tasks	
PQM 4.9.1	Prepare and Publish Deliverable Statement(s) of Intent (DSOIs) to Client
PQM 4.9.2	Review Phase DSOIs with Client
PQM 4.9.3	Document acceptance requirements for each deliverable
PQM 4.9.4	Document client and internal requirements for phase close-out
PQM 4.9.5	Schedule internal review(s) of phase deliverables
PQM 4.9.6	Schedule client review(s) of phase deliverables
PQM 4.9.7	Prepare checklist of acceptance requirements for each deliverable
PQM 4.9.8	Prepare checklist of phase close-out requirements
PQM 4.9.9	Inventory the phase deliverables
PQM 4.9.10	Conduct internal review(s) of phase deliverables <ul style="list-style-type: none"> <li>• Review deliverable(s) against acceptance requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up internal review, if needed</li> </ul>

Tasks	
PQM 4.9.11	Conduct client review(s) of phase deliverables <ul style="list-style-type: none"> <li>Review deliverable(s) against acceptance requirements</li> <li>Identify and document compliance issues, if any</li> <li>Identify and document remediation actions, if needed</li> <li>Schedule follow-up client review, if needed</li> <li>Document review results and publish to client</li> </ul>
PQM 4.9.12	Deliver phase deliverables to client for approval and sign-off
PQM 4.9.13	Collect client sign-off on phase deliverables
PQM 4.9.14	Schedule internal review of phase close-out requirements
PQM 4.9.15	Conduct internal review of phase close-out requirements <ul style="list-style-type: none"> <li>Review status of phase deliverables against close-out requirements</li> <li>Review status of remaining close-out requirements</li> <li>Identify and document compliance issues, if any</li> <li>Identify and document remediation actions, if needed</li> <li>Schedule follow-up internal review, if needed</li> <li>If ready, complete phase close-out</li> </ul>
PQM 4.9.16	Collect and document Lessons Learned

### Deliverable

A **Deliverable Statement of Intent** is prepared for each deliverable to be produced and approved in the phase. The DSOI describes the planned deliverable in sufficient detail that the client is able to evaluate the planned organization and content of the deliverable, and comment on the suitability of the planned deliverable. Achieving this agreement early in the phase is critical to deliverable acceptance later in the phase.

**Lessons Learned** are documented and collected at the end of each project phase, highlighting activities in need of adjustment or improvement to meet project objectives. Separate sessions are held for the internal project team, and the project team leadership and client participants of the project phase.

### Responsibilities

R – Project Manager

A – Project client

S – All project team members

I – Client contracting officer

## Exit Criteria

Exit Criteria include certain deliverables which must be reviewed and approved (internally and with the client) before the project may proceed to the next phase. In addition to deliverables defined and established in the method(s) that will be engaged by project(s), as well as the method(s) used by the client and those specified in the project contract, the following deliverables must be completed, reviewed, and approved before this phase is completed:

- **Database Design**, describing the major components of the data store(s) to be created and managed by the project. Administration functions and web services to be developed are also described.
- **ETL Design**, describing each of the data movement subsystems, as well as the individual ETL processes. Includes design of ETL process scheduling, logging, completion, verification of data delivery to the EDW, process restart, error data logging, data rejection logging, and notification. Administration functions and web services to be developed are also described.
- **BI Design**, describing the overall management of the business intelligence environment, as well as the individual reports, dashboards, alerts, and analytics to be delivered by the project. Administration functions and web services to be developed are also described.
- **Application Program Design**, describing the overall management of the application programs, as well as the individual data entry screens, data presentation screens, and other application functions to be delivered by the project. Administration functions and web services to be developed are also described.
- **Webpage and Portal Design**, describing the overall management of the web page and portal programs, as well as the individual web pages, portlets, and other webpage functions to be delivered by the project. Administration functions and web services to be developed are also described.
- **Requirements Traceability Matrix (updated)**, listing all requirements in requirements ID sequence. The RTM should indicate the designs (design type and section) and test cases (test case IDs) that address a requirement.
- **Physical Data Model(s)**, describing the physical organization of data in each of the data stores to be created and managed by the project.
- **Data Dictionary**, describing the business definition of all in-scope data elements to be used by the project.
- **Architecture Design (updated)**, including business, data, applications, security, and technology architecture designs and models.
- **Work Breakdown Structure (updated)**, all remaining phases must be described to the level where specific time estimates are included, team members are assigned, work products are defined, and work time is recorded in the time tracking system. Any activity with a time estimate greater than 40 hours, or a duration greater than two weeks, should be broken down to the next level of detail.
- **Deliverable Schedule (updated)**, listing all internal and contractual deliverables, for the project, and current due dates.

- **Project Decision Log (updated)**, listing all project decisions made, including date of decision and all participants in the decision.
- **Project Risk Log (updated)**, listing all identified project risks, including contingency and mitigation plans (and due date and responsible parties for those plans).
- **Project Issue Log (updated)**, listing all project issues, response plans, due dates, and responsible parties for each issue.
- **Project Status Reports**, copies of all status reports delivered to the project client during the phase.

### Key Considerations

- All contract deliverables associated with this phase must be signed off by the client for the phase to be considered closed out.
- All internal deliverables associated with this phase must complete the phase review for the phase to be considered closed out.

### Task Guidance

#### **Publish Deliverable Statement(s) of Intent (DSOIs) to Client**

At the start of the project phase, prepare a **Deliverable Statement of Intent** for each deliverable to be produced and approved in the phase. A DSOI is a 1-2 page overview of the deliverable, and will typically define the deliverable format, contents, objectives and goals, reviewers, and acceptance criteria. The contents are often described in *table of contents* format, with a brief description of each section, which may be expanded or refined in the final deliverable.

#### **Prepare checklist of acceptance requirements for each deliverable**

The Project Manager, or designated project team member, will review the project contract to identify all processes, tasks, deliverables, and work products that are to be produced during this phase of the project. All identified items are to be included in the checklist of acceptance requirements, for review during this activity.

**PQM 5.1 Educate IV&V Team**

**Objective**

The objective of this activity is to introduce a client’s IV&V Team to the scope, schedule, and testing plan for the project. The timing of this meeting will vary, but early in the Construction phase is recommended.



**Phase**

This activity is performed in the Construction phase.

Tasks	
PQM 5.1.1	Project Manager requests approval to schedule a meeting with the client’s Independent Validation & Verification (IV&V) Team
PQM 5.1.2	Project Manager schedules a meeting with the client’s IV&V Team
PQM 5.1.3	The project team assembles materials to be presented to the IV&V Team, including: <ul style="list-style-type: none"> <li>• Overview of the system test plan deliverable, including schedule</li> <li>• Overview of the application architecture, including major components and data flow</li> <li>• Overview of the technology architecture, particularly focusing on any new technologies that will be installed or upgraded</li> <li>• High-level walkthrough of the installation process</li> </ul>
PQM 5.1.4	Conduct the training for the IV&V Team, noting requests for additional information
PQM 5.1.5	Follow-up on requests for additional information

CONSTRUCTION PHASE

**Deliverable**

The **IV&V Presentation** deliverable highlights the design, quality management, and installation process for the project.

## Responsibilities

R – Technical Team Lead

A – Quality Management Supervisor/Manager, Project Manager

S – Application Designers/Developers, Quality Management Specialists

## Key Considerations

- A follow-up meeting will be scheduled early in the Testing phase, to reaffirm the project schedule and specific testing plans that will be conducted by the project team. Training in the installation process for critical components is recommended during this meeting.

**PQM 5.2 Define Test Scripts**

**Objective**

The objective of this activity is to prepare a test script for each software module (unit tests), application program set (integration tests), business transaction (system tests), and service level agreement (performance tests) to be tested. These materials include guidelines for executing the test script, specification of test data, procedures for recording results, and a test results summary report.



**Phase**

This activity is performed in the Development/Construction phase.

CONSTRUCTION PHASE

Tasks	
PQM 5.2.1	Review Requirements, Architecture Design, Application Design, and Test Case deliverables
PQM 5.2.2	For each script, assemble the testing techniques needed to verify compliance
PQM 5.2.3	Assemble test scripts -- list requirements to be tested, test data, steps to be performed, and expected results
PQM 5.2.4	Assemble test scripts into Test Scripts document, organized by Unit Tests, Integration Tests, System Tests, and Performance Tests
PQM 5.2.5	Submit draft Test Scripts deliverable to Project Manager
PQM 5.2.6	Schedule review of draft Test Scripts deliverable with project team
PQM 5.2.7	Review, update, and deliver the Test Scripts deliverable

**Deliverable**

The **Test Scripts** deliverable is a documented series of test scripts, listing the tests to be performed against each collection of functionality.

**Responsibilities**

- R – Quality Management Specialist
- A – Quality Management Supervisor/Manager, Project Manager

S – Business Analyst

C – Application Designers/Developers

I – Technical Team Lead

### Key Considerations

- Interim versions of this deliverable will be shared with efforts to create automated test scripts.

**PQM 5.3 Develop Automated Test Scripts**

**Objective**

The objective of this activity is to prepare automated test scripts for each software module (unit tests), application program set (integration tests), business transaction (system tests), and service level agreement (performance tests) to be tested using the automated testing software.



**Phase**

This activity is performed in the Development/Construction phase.

Tasks	
PQM 5.3.1	Review Test Scripts deliverable
PQM 5.3.2	Verify and update the list of scripts to be automated
PQM 5.3.3	Develop test scripts; execute and verify test scripts' operation
PQM 5.3.4	Assemble test scripts into sets of testing execution plans
PQM 5.3.5	Organize software build submission process (as described in System Testing Plan deliverable)
PQM 5.3.6	Execute tests against nightly builds <ul style="list-style-type: none"> <li>• Collect and analyze results</li> <li>• Report defects to defect tracking tool</li> <li>• Report build testing results summary to Project Manager</li> </ul>

CONSTRUCTION PHASE

**Deliverable**

The **Automated Test Scripts** deliverable is an organized library of test scripts, which can be executed against each collection of functionality.

**Responsibilities**

R – Quality Management Specialist

A – Quality Management Supervisor/Manager, Project Manager

S – Business Analyst

C – Application Designers/Developers

I – Technical Team Lead

### Key Considerations

- No considerations yet defined

**PQM 5.9 Execute Development/Construction Phase Deliverable Review**

**Objective**

The objective of this activity is to conduct the processes that review the quality of the phase work products, verifying that the project deliverables meet the defined requirements and client expectations, achieve client acceptance, and close-out the project phase.



**Phase**

This activity is performed in the Development/Construction phase.

Tasks	
PQM 5.9.1	Prepare and Publish Deliverable Statement(s) of Intent (DSOIs) to Client
PQM 5.9.2	Review Phase DSOIs with Client
PQM 5.9.3	Document acceptance requirements for each deliverable
PQM 5.9.4	Document client and internal requirements for phase close-out
PQM 5.9.5	Schedule internal review(s) of phase deliverables
PQM 5.9.6	Schedule client review(s) of phase deliverables
PQM 5.9.7	Prepare checklist of acceptance requirements for each deliverable
PQM 5.9.8	Prepare checklist of phase close-out requirements
PQM 5.9.9	Inventory the phase deliverables
PQM 5.9.10	Conduct internal review(s) of phase deliverables <ul style="list-style-type: none"> <li>• Review deliverable(s) against acceptance requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up internal review, if needed</li> </ul>

CONSTRUCTION PHASE

Tasks	
PQM 5.9.11	Conduct client review(s) of phase deliverables <ul style="list-style-type: none"> <li>Review deliverable(s) against acceptance requirements</li> <li>Identify and document compliance issues, if any</li> <li>Identify and document remediation actions, if needed</li> <li>Schedule follow-up client review, if needed</li> <li>Document review results and publish to client</li> </ul>
PQM 5.9.12	Deliver phase deliverables to client for approval and sign-off
PQM 5.9.13	Collect client sign-off on phase deliverables
PQM 5.9.14	Schedule internal review of phase close-out requirements
PQM 5.9.15	Conduct internal review of phase close-out requirements <ul style="list-style-type: none"> <li>Review status of phase deliverables against close-out requirements</li> <li>Review status of remaining close-out requirements</li> <li>Identify and document compliance issues, if any</li> <li>Identify and document remediation actions, if needed</li> <li>Schedule follow-up internal review, if needed</li> <li>If ready, complete phase close-out</li> </ul>
PQM 5.9.16	Collect and document Lessons Learned

CONSTRUCTION PHASE

### Deliverable

A **Deliverable Statement of Intent** is prepared for each deliverable to be produced and approved in the phase. The DSOI describes the planned deliverable in sufficient detail that the client is able to evaluate the planned organization and content of the deliverable, and comment on the suitability of the planned deliverable. Achieving this agreement early in the phase is critical to deliverable acceptance later in the phase.

**Lessons Learned** are documented and collected at the end of each project phase, highlighting activities in need of adjustment or improvement to meet project objectives. Separate sessions are held for the internal project team, and the project team leadership and client participants of the project phase.

### Responsibilities

R – Project Manager

A – Project client

S – All project team members

I – Client contracting officer

## Exit Criteria

Exit Criteria include certain deliverables which must be reviewed and approved (internally and with the client) before the project may proceed to the next phase. In addition to deliverables defined and established in the method(s) that will be engaged by project(s), as well as the method(s) used by the client and those specified in the project contract, the following deliverables must be completed, reviewed, and approved before this phase is completed:

- **Code Review Action Items**, describing the results of code review sessions scheduled throughout the Construction phase, including action items, responsible party, and due dates.
- **Requirements Traceability Matrix (updated)**, listing all requirements in requirements ID sequence. The RTM should indicate the test script (test script IDs) that address a requirement.
- **Work Breakdown Structure (updated)**, all remaining phases must be described to the level where specific time estimates are included, team members are assigned, work products are defined, and work time is recorded in the time tracking system. Any activity with a time estimate greater than 40 hours, or a duration greater than two weeks, should be broken down to the next level of detail.
- **Deliverable Schedule (updated)**, listing all internal and contractual deliverables, for the project, and current due dates.
- **Project Decision Log (updated)**, listing all project decisions made, including date of decision and all participants in the decision.
- **Project Risk Log (updated)**, listing all identified project risks, including contingency and mitigation plans (and due date and responsible parties for those plans).
- **Project Issue Log (updated)**, listing all project issues, response plans, due dates, and responsible parties for each issue.
- **Project Status Reports**, copies of all status reports delivered to the project client during the phase.

## Key Considerations

- All contract deliverables associated with this phase must be signed off by the client for the phase to be considered closed out.
- All internal deliverables associated with this phase must complete the phase review for the phase to be considered closed out.
- Schedule delays encountered in Development phase activities, Unit Testing, and/or Unit Tested Release will not be justification for changing the System Test Plan, nor its scope, activities, schedule, or effort commitments. Once Development phase activities commence, any proposed changes to the Testing phase schedule must be approved by the Program Manager and/or the CyberData President before they are published and used.

## Task Guidance

### **Publish Deliverable Statement(s) of Intent (DSOIs) to Client**

At the start of the project phase, prepare a **Deliverable Statement of Intent** for each deliverable to be produced and approved in the phase. A DSOI is a 1-2 page overview of the deliverable, and will typically define the deliverable format, contents, objectives and goals, reviewers, and acceptance criteria. The contents are often described in *table of contents* format, with a brief description of each section, which may be expanded or refined in the final deliverable.

### **Prepare checklist of acceptance requirements for each deliverable**

The Project Manager, or designated project team member, will review the project contract to identify all processes, tasks, deliverables, and work products that are to be produced during this phase of the project. All identified items are to be included in the checklist of acceptance requirements, for review during this activity.

**PQM 6.1 Execute Unit Testing**

**Objective**

The objective of this activity is to unit test the software modules to verify that they meet the client's requirements. Functional unit tests verify the proper operation of each requirement that is documented in the project's Requirements deliverable.



**Phase**

This activity is performed in the Testing phase.

Tasks	
PQM 6.1.1	Review project quality metrics established for Unit Testing in the System Test Plan
PQM 6.1.2	Communicate the Unit Testing quality metrics, process, and schedule to the project team
PQM 6.1.3	Developers review the Unit Testing Scripts to understand the testing requirements for their assigned software module(s)
PQM 6.1.4	Prepare the Unit Testing environment
PQM 6.1.5	Developers test their software module(s): <ul style="list-style-type: none"> <li>• checking off completed tests</li> <li>• documenting defects identified</li> <li>• documenting remediation actions</li> </ul>
PQM 6.1.6	Developers repair defects identified and repeat the test(s) of their software modules
PQM 6.1.7	Developers report all Unit Testing status and results through the defect management and reporting tool
PQM 6.1.8	Regularly review defect management progress with project team
PQM 6.1.9	Report actual defect statistics against Unit Testing quality metrics to the project team
PQM 6.1.10	When the project's Unit Testing quality metrics have been achieved, assemble the Unit Testing Results deliverable
PQM 6.1.11	Submit draft Unit Testing Results deliverable to Project Manager
PQM 6.1.12	Schedule review of draft Unit Testing Results deliverable with project team

Tasks	
PQM 6.1.13	Review, update, and deliver the Unit Testing Results deliverable

### Deliverable

The **Unit Testing Results** deliverable documents the number of unit tests conducted, the number of documented unit tests, and the number of successes and failures. The results also provide details about code changes that are made to resolve any issues highlighted by failed test cases. The Unit Testing Results deliverable includes a report of actual defect statistics against Unit Testing quality metrics.

### Responsibilities

R – Technical Team Lead

A – Quality Management Supervisor/Manager, Project Manager

S – Application Designers/Developers \*

\* Time permitting, Business Analysts and Quality Management Specialists will also participate in these Unit Testing tasks. However, the tasks performed in this Testing activity continue to be the responsibility of the Application Designers/Developers.

### Key Considerations

- It is a requirement of the Execute Unit Testing activity to verify that all documented requirements are met by the application system. Downstream testing activities will select a sampling of requirements for spot-checking. If defect thresholds for any of those activities are exceeded, the project may be reverted back to re-execute the Execute Unit Testing activity.
- Downstream testing activities may indicate that sufficient repairs need to be made to one or more software modules, to the extent that Unit Testing for the software module(s) must be performed again before the downstream testing activities may resume.
- Reporting actual defect statistics may be a regularly occurring event (weekly, daily) until the Unit Testing quality metrics are achieved.
- Schedule delays encountered in Development phase activities, Unit Testing, and/or Unit Tested Release will not be justification for changing the System Test Plan, nor its scope, activities, schedule, or effort commitments. Once Development phase activities commence, any proposed changes to the Testing phase schedule must be approved by the Program Manager and/or the CyberData President before they are published and used.
- In the Definition phase, the client identifies their quality metric requirements, which describe the minimum quality metrics for an acceptable delivery of the project's

deliverable(s). In the Requirements phase, the System Test Plan describes the minimal metrics for each testing activity, with the final activity delivering to the client's quality metric requirements. The Execute Unit Testing activity is not considered complete until the unit tests, across all software components, meet or exceed the minimum metrics established for Unit Testing.

### **Defect Reporting and Remediation**

The project team will document software defects, as they are identified and during all Testing phase activities, in the CyberData-designated defect tracking system, currently Bugzilla. During the Execute Unit Testing activity, each developer is responsible for logging all defects that they detect in their work products, as well as other work products that they are testing. The project's Quality Management Specialist(s) will log all defects detected in the Integration, System, and UAT activities. Each defect will be assigned a severity level, description, identification date, assigned party, and a status.

Performance metrics will be tracked, including time to resolution, number of defects (identified by Testing activity, application component, work product author, and other categories), and current status. Quality metrics will be established for each Testing activity, requiring successful achievement before the project progresses to the next activity.

Defects are to be repaired by the author of the work product (software component or document). Quality Management Specialists are not to make changes to others' work products.

The defect tracking system's project dashboard provides up-to-the-minute status on the project's recorded defects. The Project Manager will regularly review status with the project team to update progress, prioritize assignments, and brainstorm defect resolution techniques. The Project Manager will regularly report information about the testing and defect management status to the client Project Manager.

As defects are repaired, the assigned party will update the defect tracking system to indicate that the work product is ready for review and/or testing. During Unit Testing, the application developer will retest the application component, verifying that all unit tests are completed successfully. In later Testing activities, the application developer will submit the new version of remediated software component to the CyberData version control system. The defect will be retested, and the defect status updated accordingly.

**PQM 6.2 Build Release Candidate**

**Objective**

The objective of this activity is to prepare, package, and deliver a full build of the developed software for evaluation in the Testing environment. A Build Plan is prepared, providing the Development Team with guidance on how to assemble the Release Candidate.



**Phase**

This activity is performed in the Testing phase.

Tasks	
PQM 6.2.1	Identify documentation and build packaging requirements
PQM 6.2.2	Quality Management Specialist assembles and prepares the Build Evaluation Checklist
PQM 6.2.3	Collect reference documentation needed to prepare the Release Candidate
PQM 6.2.4	Assemble list of known defects, corrected defects, and current quality metric measures
PQM 6.2.5	Conduct Build Plan meeting: <ul style="list-style-type: none"> <li>• generate Bugzilla summary report</li> <li>• prepare list of new features</li> <li>• prepare list of repaired defects</li> <li>• prepare list of components and sources for each</li> </ul>
PQM 6.2.6	Prepare Build Plan
PQM 6.2.7	Development Team reviews plan
PQM 6.2.8	Project Manager approves Build Plan
PQM 6.2.9	Project Manager emails Build Plan to Project Managers and project’s Quality Management Specialist
PQM 6.2.10	Quality Management Specialist updates list of scheduled builds
PQM 6.2.11	Prepare installation, de-installation, and installation validation procedures
PQM 6.2.12	Prepare supporting documentation

Tasks	
PQM 6.2.13	Prepare Build Execution Checklist, listing all actions (except where already detailed by the Installation Procedures) to be performed to achieve a validated installation of the build
PQM 6.2.14	Check documentation and software components into version control system's shared repository
PQM 6.2.15	Uniquely number the build
PQM 6.2.16	Assemble documentation and software components for the build
PQM 6.2.17	Burn the Release Candidate to CD(s)
PQM 6.2.18	Project Manager verifies that the submission of project work products is complete; hands over Release Candidate to Quality Management Specialist
PQM 6.2.19	The Quality Management Specialist checks all of the CD's build components into VSS (the secure QA area)
PQM 6.2.20	Verify Release Candidate
PQM 6.2.21	Quality Management Specialist schedules installation of the new build into the QA environment
PQM 6.2.22	Quality Management Specialist installs the new Release Candidate, following the Build Execution Checklist, and performs the validation process included in the Installation Procedures

### Deliverable

The **Release Candidate** is a *functionally complete, verifiable, documented, and demonstrable* build that represents what the Development Team certifies as a potential release to post-unit testing activities and/or Production.

The Release Candidate deliverable is a CD copy of all release documentation and software work products, secured in the version control system. The Project Manager will verify that all required functionality, as documented in the Requirements Traceability Matrix (RTM), is included in the Release Candidate. The Project Manager will verify that all documentation and software components needed to install and test the build, including scripts and configuration files, are included in the Release Candidate CD(s). The Release Candidate will be used to prepare the Testing environment for Integration, System, UAT, and Performance Testing.

The Build Plan describes the contents of the build, instructing the team members in its assembly. The Build Plan should highlight the objectives of the build (deliver new functionality, repair defects, etc.), and provide a descriptive list of the major new capabilities and repaired defects.

## Responsibilities

R – Technical Team Lead

A – Quality Management Supervisor/Manager, Project Manager

S – Application Designers/Developers, Business Analyst, Quality Management Specialist

## Key Considerations

- The lack of a Build Plan has been determined to be a significant contributor to poor quality in a build – selecting wrong components, missing components, wrong versions of components, and more can result without an agreed-upon plan for creating the build.
- Unlike most activities in this method, the Build Release Candidate activity is likely to be executed multiple times during the Testing phase.
- While project work products may be checked into the version control system in a sequence of steps or stages, this activity will not be considered complete until all project work products have been checked in and the Project Manager produces a new build.
- Schedule delays encountered in Development phase activities, Unit Testing, and/or Build Release Candidate activities will not be justification for changing the System Test Plan, nor its scope, activities, schedule, or effort commitments. Once Development phase activities commence, any proposed changes to the Testing phase schedule must be approved by the Program Manager and/or the CyberData President before they are published and used.

## Task Guidance

### **Prepare installation, de-installation, and installation validation procedures**

The Installation Procedures document will include a current list of all documentation and software components included in the Release Candidate. De-installation Procedures will describe the actions to be taken to restore the system to its pre-installation state. Installation Validation Procedures describe the actions that need to be taken to verify that the Installation Procedures were properly performed. Be sure that the person that is to execute the Installation Validation Procedures has the authority and system account privileges to perform the validation procedures.

### **Prepare Build Plan**

The **Build Plan** describes the contents of the build, instructing the team members in its assembly. The Build Plan should highlight the objectives of the build (deliver new functionality, repair defects, etc.), provide a descriptive list of the major new capabilities and repaired defects, list the software components of the build and where they are sourced from, list the

documentation components of the build and where they are sourced from, and list any dependencies (such as other builds) that must be installed prior to the build.

The Build Plan document should contain the following sections:

- **Overview/Purpose** – A business description of what the build provides to the client
- **New Features** – For a small number of features, list the business requirements (including business requirement IDs); For a larger list, summarize the major new features.
- **Repaired Defects** – List the defects repaired in this release (including defect IDs from the defect tracking system).
- **Dependencies** – List the prior builds, including release numbers, that must be in place before this build is installed. Consider environmental, documentation, and data refresh builds.
- **Build Components** – List the software components that must be included in the build. Identify where the build assembler will find these components.
- **Documentation Components** – List the documents that must be revised and included in the build.
- **Quality Assurance Guidance** – Any special instructions or guidance that Quality Assurance needs to perform installation and validation procedures

The Build Plan document should have a cover page, document history, and other pages normally found in documentation prepared for the client.

### **Burn the Release Candidate to CD(s)**

This task prepares a complete copy of the unit tested system. Besides prerequisites, which are listed in the Installation Procedures, no other documentation or software is required to install and verify the build.

### **Project Manager verifies that the submission of project work products is complete; hands over Release Candidate to Quality Management Specialist**

At this point in the Software Development Lifecycle (SDLC), the QA team owns the Candidate Release and the Development Team is not permitted to make any changes.

### **Verify Release Candidate**

The Quality Management Specialist verifies the version control system's contents for the Release Candidate match what is documented in the Build Plan and in the Installation Procedures. Quality Management Specialist verifies that the documentation contents of the build meet the documentation requirements for those contents, according to the Build Evaluation Checklist

### **Quality Management Specialist installs the new Release Candidate**

Before installing the new Release Candidate, the Quality Assurance testing environment should be configured identically to, or certifiably mimics, the client's Production environment.



**No New or Revised Functionality**

Once the project completes this activity for the first time, no new or revised functionality may be introduced to the project. No changes in design or new programs (including data feeds and reports) may be introduced. Only repairs to known, documented defects (that are discovered during downstream testing activities) will be accepted by the project's Quality Management Specialist.

**PQM 6.3 Update IV&V Team**

**Objective**

The objective of this activity is to provide the client’s IV&V Team with the necessary information that allows them to complete their preparations for testing. The timing of this meeting will vary, but earliest in the Testing phase is recommended.



**Phase**

This activity is performed in the Testing phase.

Tasks	
PQM 6.3.1	Project Manager requests approval to schedule a meeting with the client’s Independent Validation & Verification (IV&V) Team
PQM 6.3.2	Project Manager schedules a meeting with the client’s IV&V Team
PQM 6.3.3	The project team assembles materials to be presented to the IV&V Team, including: <ul style="list-style-type: none"> <li>• the project Requirements document</li> <li>• the project Installation Procedures</li> <li>• an updated presentation of the project team’s quality management activities and schedule</li> <li>• a recommended list of contacts and contact procedures available to the IV&amp;V Team during installation and testing tasks</li> <li>• any required training in the project application and/or installation procedures</li> </ul>
PQM 6.3.4	Conduct the meeting with the IV&V Team, noting requests for additional information
PQM 6.3.5	Follow-up on requests for additional information

**Deliverable**

The **IV&V Final Presentation** deliverable includes the Requirements document and Installation Procedures, overall quality management plan and remaining schedule, contact list, and any remaining installation process for the project.

TESTING PHASE

## Responsibilities

R – Technical Team Lead

A – Quality Management Supervisor/Manager, Project Manager

S – Application Designers/Developers, Quality Management Specialists

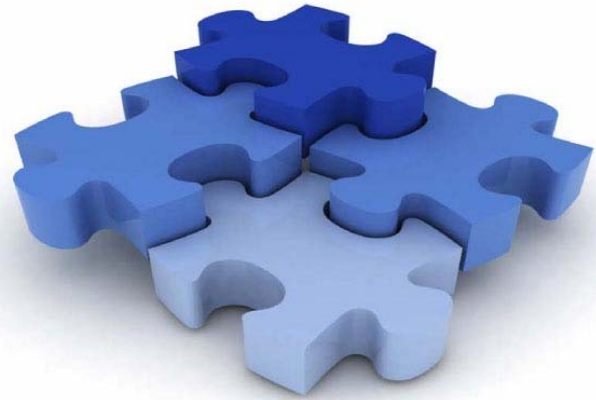
## Key Considerations

- No considerations identified at this time.

**PQM 6.4 Execute Integration Testing**

**Objective**

The objective of this activity is to conduct the integration tests, as described in the Integration Test Cases and Integration Test Scripts deliverables. Testing occurs on the prepared testing environment and is a well-orchestrated replication of system and business processes.



**Phase**

This activity is performed in the Testing phase.

Tasks	
PQM 6.4.1	Review project quality metrics established for Integration Testing in the System Test Plan
PQM 6.4.2	Communicate the Integration Testing quality metrics, process, and schedule to the project team
PQM 6.4.3	Integration test participants review the Integration Testing Scripts to understand the testing requirements for their assigned application program set(s)
PQM 6.4.4	Prepare the Integration Testing environment, including sample data
PQM 6.4.5	Integration test participants test their assigned application program set(s) <ul style="list-style-type: none"> <li>• checking off completed tests</li> <li>• documenting defects identified</li> <li>• documenting remediation actions</li> </ul>
PQM 6.4.6	Developers repair defects identified and resubmit their software module(s) for retesting
PQM 6.4.7	Integration test participants report all Integration Testing results to the Quality Management Specialist
PQM 6.4.8	Document outstanding defects in the defect management and reporting tool
PQM 6.4.9	Report actual defect statistics against Integration Testing quality metrics to the project team

TESTING PHASE

Tasks	
PQM 6.4.10	When the project’s Integration Testing quality metrics have been achieved, assemble the Integration Testing Results deliverable
PQM 6.4.11	Submit draft Integration Testing Results deliverable to Project Manager
PQM 6.4.12	Schedule review of draft Integration Testing Results deliverable with project team
PQM 6.4.13	Review, update, and deliver the Integration Testing Results deliverable

### Deliverable

At the end of the Execute Integration Testing activity, all planned integration test cases/scripts will have been run, and any defects causing test case failures documented in the Defect Report and Status Report. The information documented in the Defect Report and Status Report should be used to generate the **Integration Testing Results**.

The Integration Testing Results is the final report artifact that summarizes all that was completed during the test activity. The report lists the documented test cases performed, the count of each test case conducted, and the number of successes and failures. It documents the defects found, defects resolved, along with any remaining open issues. The results also provide details about code changes that are made to resolve any issues highlighted by failed test cases. A listing of all regression tests performed and the resolved defects during the activity is included. The report summarizes the project’s performance during the test activity and includes Test Execution and Pass Rate Analysis. The Integration Testing Results deliverable includes a report of actual defect statistics against Integration Testing quality metrics.

### Responsibilities

- R – Quality Management Specialist
- A – Quality Management Supervisor/Manager, Project Manager
- S – Application Designers/Developers, Business Analyst
- I – Technical Team Lead

### Key Considerations

- Testing activities may indicate that sufficient repairs need to be made to one or more software modules, to the extent that Unit Testing and Integration Testing for the software module(s) must be performed again before the testing activities may resume.
- Reporting actual defect statistics may be a regularly occurring event (weekly, daily) until the Integration Testing quality metrics are achieved.
- It is a requirement of the Execute Unit Testing activity to verify that all documented requirements are met by the application system. Downstream testing activities, such as

Integration Testing, will select a sampling of requirements for spot-checking. If defect thresholds for any of those activities are exceeded, the project may be reverted back to re-execute the Execute Unit Testing activity.

- In the Definition phase, the client identifies their quality metric requirements, which describe the minimum quality metrics for an acceptable delivery of the project's deliverable(s). In the Requirements phase, the System Test Plan describes the minimal metrics for each testing activity, with the final activity delivering to the client's quality metric requirements. The Execute Integration Testing activity is not considered complete until the integration tests meet or exceed the minimum metrics established for Integration Testing.

## Task Guidance

### **Prepare the Integration Testing environment, including sample data**

Test data sets consist of groups of data used to verify specific functionality. The test data sets should be developed by the Quality Management team as well as the Development team. The test data sets must contain a sample of every category of valid data as well as many invalid conditions as possible. Data sets can include such values as bounds checking, negative data, min / max values and null data. These sets are used to verify the specific transformation between specific modules for Integration testing.

**PQM 6.5 Execute System Testing**

**Objective**

The objective of this activity is to conduct the system tests, as described in the System Test Cases and System Test Scripts deliverables. Testing occurs on the prepared testing environment and is a well-orchestrated replication of system and business processes.



**Phase**

This activity is performed in the Testing phase.

Tasks	
PQM 6.5.1	Review project quality metrics established for System Testing in the System Test Plan
PQM 6.5.2	Communicate the System Testing quality metrics, process, and schedule to the project team
PQM 6.5.3	System test participants review the System Testing Scripts to understand the testing requirements for their assigned business transactions and processes
PQM 6.5.4	Prepare the System Testing environment, including sample data
PQM 6.5.5	System test participants test their assigned business transactions and processes <ul style="list-style-type: none"> <li>• checking off completed tests</li> <li>• documenting defects identified</li> <li>• documenting remediation actions</li> </ul>
PQM 6.5.6	Developers repair defects identified and resubmit their software module(s) for retesting
PQM 6.5.7	System test participants report all System Testing results to the Quality Management Specialist
PQM 6.5.8	Document outstanding defects in the defect management and reporting tool
PQM 6.5.9	Report actual defect statistics against System Testing quality metrics to the project team
PQM 6.5.10	When the project's System Testing quality metrics have been achieved, assemble the System Testing Results deliverable

TESTING PHASE

Tasks	
PQM 6.5.11	Submit draft System Testing Results deliverable to Project Manager
PQM 6.5.12	Schedule review of draft System Testing Results deliverable with project team
PQM 6.5.13	Review, update, and deliver the System Testing Results deliverable

### Deliverable

At the end of the Execute System Testing activity, all planned system test cases/scripts will have been run, and any defects causing test case failures documented in the Defect Report and Status Report. The information documented in the Defect Report and Status Report should be used to generate the **System Testing Results**.

The System Testing Results is the final report artifact that summarizes all that was completed during the test activity. The report lists the documented test cases performed, the count of each test case conducted, and the number of successes and failures. It documents the defects found, defects resolved, along with any remaining open issues. The results also provide details about code changes that are made to resolve any issues highlighted by failed test cases. A listing of all regression tests performed and the resolved defects during the activity is included. The report summarizes the project’s performance during the test activity and includes Test Execution and Pass Rate Analysis. The System Testing Results deliverable includes a report of actual defect statistics against System Testing quality metrics.

### Responsibilities

- R – Quality Management Specialist
- A – Quality Management Supervisor/Manager, Project Manager
- S – Application Designers/Developers, Business Analyst
- I – Technical Team Lead

### Key Considerations

- It is a requirement of the Execute Unit Testing activity to verify that all documented requirements are met by the application system. Downstream testing activities, such as System Testing, will select a sampling of requirements for spot-checking. If defect thresholds for any of those activities are exceeded, the project may be reverted back to re-execute the Execute Unit Testing activity.
- Testing activities may indicate that sufficient repairs need to be made to one or more software modules, to the extent that upstream testing must be performed again before the testing activities may resume.
- Reporting actual defect statistics may be a regularly occurring event (weekly, daily) until the System Testing quality metrics are achieved.

- In the Definition phase, the client identifies their quality metric requirements, which describe the minimum quality metrics for an acceptable delivery of the project's deliverable(s). In the Requirements phase, the System Test Plan describes the minimal metrics for each testing activity, with the final activity delivering to the client's quality metric requirements. The Execute System Testing activity is not considered complete until the system tests meet or exceed the minimum metrics established for System Testing.

**PQM 6.6 Execute User Acceptance Testing**

**Objective**

The objective of this activity is to conduct the functional and transaction tests, as described in the User Acceptance Test Cases and User Acceptance Test Scripts deliverables. Testing occurs on the prepared testing environment and is conducted by a select set of the client organization’s subject matter experts.



**Phase**

This activity is performed in the Testing phase.

Tasks	
PQM 6.6.1	Schedule preparatory meeting with UAT participants
PQM 6.6.2	Schedule preparatory meeting with project team
PQM 6.6.3	Review project quality metrics established for User Acceptance Testing in the System Test Plan
PQM 6.6.4	Conduct preparatory meeting with UAT participants <ul style="list-style-type: none"> <li>• Present the UAT quality metrics, process, and schedule</li> <li>• Present sample User Acceptance tests</li> <li>• Distribute copies of the User Acceptance Test Cases and User Acceptance Test Scripts to UAT participants</li> </ul>
PQM 6.6.5	Conduct preparatory meeting with project team <ul style="list-style-type: none"> <li>• Present the UAT quality metrics, process, and schedule</li> <li>• Present sample User Acceptance tests</li> <li>• Present roles and responsibilities for supporting project personnel</li> </ul>
PQM 6.6.6	Prepare the User Acceptance Testing environment, including sample data
PQM 6.6.7	UAT participants test their assigned business transactions and processes <ul style="list-style-type: none"> <li>• checking off completed tests</li> <li>• documenting defects identified</li> </ul>
PQM 6.6.8	Developers repair defects identified, document remediation efforts performed, and resubmit their software module(s) for retesting
PQM 6.6.9	UAT participants report all UAT results to the Quality Management Specialist

Tasks	
PQM 6.6.10	Document outstanding defects in the defect management and reporting tool
PQM 6.6.11	Report actual defect statistics against UAT quality metrics to the UAT participants and the project team
PQM 6.6.12	When the project's UAT quality metrics have been achieved, assemble the User Acceptance Testing Results deliverable
PQM 6.6.13	Submit draft User Acceptance Testing Results deliverable to Project Manager
PQM 6.6.14	Schedule review of draft User Acceptance Testing Results deliverable with project team
PQM 6.6.15	Review, update, and deliver the User Acceptance Testing Results deliverable

### Deliverable

At the end of the Execute User Acceptance Testing activity, all planned user acceptance test cases/scripts will have been run, and any defects causing test case failures documented in the Defect Report and Status Report. The information documented in the Defect Report and Status Report should be used to generate the **User Acceptance Testing Results**.

The User Acceptance Testing Results is the final report artifact that summarizes all that was completed during the test activity. The report lists the documented test cases performed, the count of each test case conducted, and the number of successes and failures. It documents the defects found, defects resolved, along with any remaining open issues. The results also provide details about code changes that are made to resolve any issues highlighted by failed test cases. A listing of all regression tests performed and the resolved defects during the activity is included. The report summarizes the project's performance during the test activity and includes Test Execution and Pass Rate Analysis. The User Acceptance Testing Results deliverable includes a report of actual defect statistics against User Acceptance Testing quality metrics.

### Responsibilities

R – Quality Management Specialist

A – Quality Management Supervisor/Manager, Project Manager

S – Client Participants, Application Designers/Developers, Business Analyst

I – Technical Team Lead

### Key Considerations

- It is a requirement of the Execute Unit Testing activity to verify that all documented requirements are met by the application system. Downstream testing activities, such as User Acceptance Testing, will select a sampling of requirements for spot-checking. If

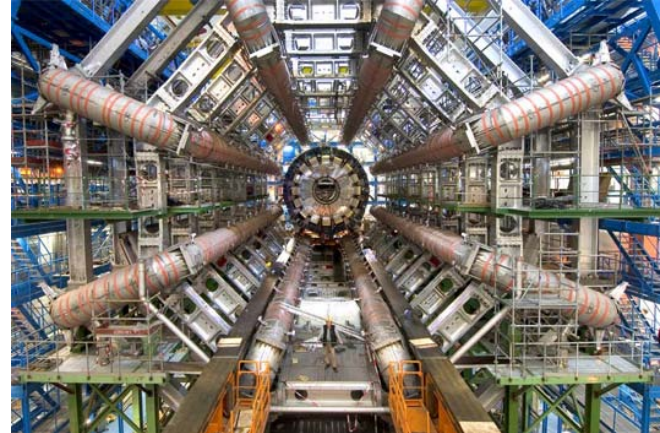
defect thresholds for any of those activities are exceeded, the project may be reverted back to re-execute the Execute Unit Testing activity.

- Testing activities may indicate that sufficient repairs need to be made to one or more software modules, to the extent that upstream testing must be performed again before the testing activities may resume.
- Reporting actual defect statistics may be a regularly occurring event (weekly, daily) until the User Acceptance Testing quality metrics are achieved.
- In the Definition phase, the client identifies their quality metric requirements, which describe the minimum quality metrics for an acceptable delivery of the project's deliverable(s). In the Requirements phase, the System Test Plan describes the minimal metrics for each testing activity, with the final activity delivering to the client's quality metric requirements. The Execute User Acceptance Testing activity is not considered complete until the user acceptance tests meet or exceed the minimum metrics established for User Acceptance Testing in the System Test Plan.

**PQM 6.7 Execute Performance Testing**

**Objective**

Performance testing evaluates the application system’s ability to deliver the established performance and scalability requirements. Performance is evaluated at escalating volumes of concurrent transaction counts. Testing occurs on the prepared testing environment and focuses on transaction performance, throughput, and identifying system bottlenecks.



**Phase**

This activity is performed in the Testing phase.

Tasks	
PQM 6.7.1	Review project quality metrics established for Performance Testing in the System Test Plan
PQM 6.7.2	Communicate the Performance Testing quality metrics, process, and schedule to the project team
PQM 6.7.3	Performance test participants review the Performance Testing Scripts to understand the testing requirements for their assigned business transactions and processes
PQM 6.7.4	Prepare the Performance Testing environment, including sample data
PQM 6.7.5	Performance test participants test their assigned business transactions and processes <ul style="list-style-type: none"> <li>• recording performance statistics</li> <li>• reporting results to the Quality Management Specialist</li> </ul>
PQM 6.7.6	Document non-compliant performance statistics in the defect management and reporting tool
PQM 6.7.7	Project team determines remediation actions required, assigning applicable project team members
PQM 6.7.8	Report actual performance statistics against Performance Testing performance metrics to the project team

TESTING PHASE

Tasks	
PQM 6.7.9	When the project's Performance Testing performance metrics have been achieved, assemble the Performance Testing Results deliverable
PQM 6.7.10	Submit draft Performance Testing Results deliverable to Project Manager
PQM 6.7.11	Schedule review of draft Performance Testing Results deliverable with project team
PQM 6.7.12	Review, update, and deliver the Performance Testing Results deliverable

### Deliverable

The **Performance Testing Results** deliverable documents the number of performance tests conducted and the number of successes and failures. The results also provide details about code changes that are made to resolve any issues highlighted by failed test cases. The Performance Testing Results deliverable includes a report of actual performance statistics against Performance Testing performance metrics.

### Responsibilities

R – Quality Management Specialist

A – Quality Management Supervisor/Manager, Project Manager

S – Application Designers/Developers, Business Analyst, Project Architect

I – Technical Team Lead

### Key Considerations

- Testing activities may indicate that sufficient repairs need to be made to one or more software modules, to the extent that upstream testing must be performed again before the testing activities may resume.
- Reporting actual defect statistics may be a regularly occurring event (weekly, daily) until the Performance Testing performance metrics are achieved.
- In the Definition phase, the client identifies their quality metric requirements, which describe the minimum quality metrics for an acceptable delivery of the project's deliverable(s). In the Requirements phase, the System Test Plan describes the minimal metrics for each testing activity, with the final activity delivering to the client's quality metric requirements. The Execute Performance Testing activity is not considered complete until the performance tests meet or exceed the minimum metrics established for Performance Testing in the System Test Plan.

**PQM 6.9 Execute Testing Phase Deliverable Review**

**Objective**

The objective of this activity is to conduct the processes that review the quality of the phase work products, verifying that the project deliverables meet the defined requirements and client expectations, achieve client acceptance, and close-out the project phase.



**Phase**

This activity is performed in the Testing phase.

Tasks	
PQM 6.9.1	Prepare and Publish Deliverable Statement(s) of Intent (DSOIs) to Client
PQM 6.9.2	Review Phase DSOIs with Client
PQM 6.9.3	Document acceptance requirements for each deliverable
PQM 6.9.4	Document client and internal requirements for phase close-out
PQM 6.9.5	Schedule internal review(s) of phase deliverables
PQM 6.9.6	Schedule client review(s) of phase deliverables
PQM 6.9.7	Prepare checklist of acceptance requirements for each deliverable
PQM 6.9.8	Prepare checklist of phase close-out requirements
PQM 6.9.9	Inventory the phase deliverables
PQM 6.9.10	Conduct internal review(s) of phase deliverables <ul style="list-style-type: none"> <li>Review deliverable(s) against acceptance requirements</li> <li>Identify and document compliance issues, if any</li> <li>Identify and document remediation actions, if needed</li> <li>Schedule follow-up internal review, if needed</li> </ul>
PQM 6.9.11	Conduct client review(s) of phase deliverables <ul style="list-style-type: none"> <li>Review deliverable(s) against acceptance requirements</li> <li>Identify and document compliance issues, if any</li> <li>Identify and document remediation actions, if needed</li> <li>Schedule follow-up client review, if needed</li> <li>Document review results and publish to client</li> </ul>

Tasks	
PQM 6.9.12	Deliver phase deliverables to client for approval and sign-off
PQM 6.9.13	Collect client sign-off on phase deliverables
PQM 6.9.14	Schedule internal review of phase close-out requirements
PQM 6.9.15	Conduct internal review of phase close-out requirements <ul style="list-style-type: none"> <li>• Review status of phase deliverables against close-out requirements</li> <li>• Review status of remaining close-out requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up internal review, if needed</li> <li>• If ready, complete phase close-out</li> </ul>
PQM 6.9.16	Collect and document Lessons Learned

### Deliverable

A **Deliverable Statement of Intent** is prepared for each deliverable to be produced and approved in the phase. The DSOI describes the planned deliverable in sufficient detail that the client is able to evaluate the planned organization and content of the deliverable, and comment on the suitability of the planned deliverable. Achieving this agreement early in the phase is critical to deliverable acceptance later in the phase.

**Lessons Learned** are documented and collected at the end of each project phase, highlighting activities in need of adjustment or improvement to meet project objectives. Separate sessions are held for the internal project team, and the project team leadership and client participants of the project phase.

### Responsibilities

R – Project Manager

A – Project client

S – All project team members

I – Client contracting officer

### Exit Criteria

Exit Criteria include certain deliverables which must be reviewed and approved (internally and with the client) before the project may proceed to the next phase. In addition to deliverables defined and established in the method(s) that will be engaged by project(s), as well as the method(s) used by the client and those specified in the project contract, the following deliverables must be completed, reviewed, and approved before this phase is completed:

- **Training Plan**, describing the training curriculum, courses, schedule(s), and participants.
- **Work Breakdown Structure (updated)**, all remaining phases must be described to the level where specific time estimates are included, team members are assigned, work products are defined, and work time is recorded in the time tracking system. Any activity with a time estimate greater than 40 hours, or a duration greater than two weeks, should be broken down to the next level of detail.
- **Deliverable Schedule (updated)**, listing all internal and contractual deliverables, for the project, and current due dates.
- **Project Decision Log (updated)**, listing all project decisions made, including date of decision and all participants in the decision.
- **Project Risk Log (updated)**, listing all identified project risks, including contingency and mitigation plans (and due date and responsible parties for those plans).
- **Project Issue Log (updated)**, listing all project issues, response plans, due dates, and responsible parties for each issue.
- **Project Status Reports**, copies of all status reports delivered to the project client during the phase.

### Key Considerations

- All contract deliverables associated with this phase must be signed off by the client for the phase to be considered closed out.
- All internal deliverables associated with this phase must complete the phase review for the phase to be considered closed out.

### Task Guidance

#### **Publish Deliverable Statement(s) of Intent (DSOIs) to Client**

At the start of the project phase, prepare a **Deliverable Statement of Intent** for each deliverable to be produced and approved in the phase. A DSOI is a 1-2 page overview of the deliverable, and will typically define the deliverable format, contents, objectives and goals, reviewers, and acceptance criteria. The contents are often described in *table of contents* format, with a brief description of each section, which may be expanded or refined in the final deliverable.

#### **Prepare checklist of acceptance requirements for each deliverable**

The Project Manager, or designated project team member, will review the project contract to identify all processes, tasks, deliverables, and work products that are to be produced during this phase of the project. All identified items are to be included in the checklist of acceptance requirements, for review during this activity.

**PQM 7.2 Educate Production Team**

**Objective**

The objective of this activity is to introduce a client's Production Team to the scope, schedule, and installation plan for the project. The timing of this meeting will vary, but early in the Testing or IV&V phase is recommended.



**Phase**

The timing of this meeting will vary, but early in the Testing or IV&V phase is recommended.

Tasks	
PQM 7.2.1	Project Manager requests approval to schedule a meeting with the client's Production Team
PQM 7.2.2	Project Manager schedules a meeting with the client's Production Team
PQM 7.2.3	<p>The project team assembles materials to be presented to the Production Team, including:</p> <ul style="list-style-type: none"> <li>• Overview of the system installation deliverable, including installation time requirements</li> <li>• Overview of the application architecture, including major components and data flow</li> <li>• Overview of the technology architecture, particularly focusing on any new technologies that will be installed or upgraded</li> <li>• a recommended list of contacts and contact procedures available to the Production Team during installation tasks</li> <li>• any required training in the project application and/or installation procedures</li> <li>• Copy of the Installation Procedures documentation</li> </ul>
PQM 7.2.4	Conduct the training for the Production Team, noting requests for additional information
PQM 7.2.5	Follow-up on requests for additional information

## Deliverable

The **Production Team Presentation** deliverable highlights the design, installation process, and installation support contacts for the project.

## Responsibilities

R – Technical Team Lead

A – Quality Management Supervisor/Manager, Project Manager

S – Application Designers/Developers, Quality Management Specialists

## Key Considerations

- Approximately one week prior to the official OBRR submission, the Development Team should submit an email request for a meeting with the client's Production Team. The request should include the installation manual.

**PQM 7.9 Execute IV&V Phase Deliverable Review**

**Objective**

The objective of this activity is to conduct the processes that review the quality of the phase work products, verifying that the project deliverables meet the defined requirements and client expectations, achieve client acceptance, and close-out the project phase.



**Phase**

This activity is performed in the IV&V phase.

Tasks	
PQM 7.9.1	Prepare and Publish Deliverable Statement(s) of Intent (DSOIs) to Client
PQM 7.9.2	Review Phase DSOIs with Client
PQM 7.9.3	Document acceptance requirements for each deliverable
PQM 7.9.4	Document client and internal requirements for phase close-out
PQM 7.9.5	Schedule internal review(s) of phase deliverables
PQM 7.9.6	Schedule client review(s) of phase deliverables
PQM 7.9.7	Prepare checklist of acceptance requirements for each deliverable
PQM 7.9.8	Prepare checklist of phase close-out requirements
PQM 7.9.9	Inventory the phase deliverables
PQM 7.9.10	Conduct internal review(s) of phase deliverables <ul style="list-style-type: none"> <li>• Review deliverable(s) against acceptance requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up internal review, if needed</li> </ul>
PQM 7.9.11	Conduct client review(s) of phase deliverables <ul style="list-style-type: none"> <li>• Review deliverable(s) against acceptance requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up client review, if needed</li> <li>• Document review results and publish to client</li> </ul>

Tasks	
PQM 7.9.12	Deliver phase deliverables to client for approval and sign-off
PQM 7.9.13	Collect client sign-off on phase deliverables
PQM 7.9.14	Schedule internal review of phase close-out requirements
PQM 7.9.15	Conduct internal review of phase close-out requirements <ul style="list-style-type: none"> <li>• Review status of phase deliverables against close-out requirements</li> <li>• Review status of remaining close-out requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up internal review, if needed</li> <li>• If ready, complete phase close-out</li> </ul>
PQM 7.9.16	Collect and document Lessons Learned

### Deliverable

A **Deliverable Statement of Intent** is prepared for each deliverable to be produced and approved in the phase. The DSOI describes the planned deliverable in sufficient detail that the client is able to evaluate the planned organization and content of the deliverable, and comment on the suitability of the planned deliverable. Achieving this agreement early in the phase is critical to deliverable acceptance later in the phase.

**Lessons Learned** are documented and collected at the end of each project phase, highlighting activities in need of adjustment or improvement to meet project objectives. Separate sessions are held for the internal project team, and the project team leadership and client participants of the project phase.

### Responsibilities

R – Project Manager

A – Project client

S – All project team members

I – Client contracting officer

### Exit Criteria

Exit Criteria include certain deliverables which must be reviewed and approved (internally and with the client) before the project may proceed to the next phase. In addition to deliverables defined and established in the method(s) that will be engaged by project(s), as well as the method(s) used by the client and those specified in the project contract, the following deliverables must be completed, reviewed, and approved before this phase is completed:

- **Deployment Plan**, describing the activities that will validate that the developed software, data, and user accounts, have been properly installed. This plan should include notifications to the user community about the availability of the new system.
- **Help Desk Management Plan**, describing services, services hours, contact procedures, and performance metrics.
- **Work Breakdown Structure (updated)**, all remaining phases must be described to the level where specific time estimates are included, team members are assigned, work products are defined, and work time is recorded in the time tracking system. Any activity with a time estimate greater than 40 hours, or a duration greater than two weeks, should be broken down to the next level of detail.
- **Deliverable Schedule (updated)**, listing all internal and contractual deliverables, for the project, and current due dates.
- **Project Decision Log (updated)**, listing all project decisions made, including date of decision and all participants in the decision.
- **Project Risk Log (updated)**, listing all identified project risks, including contingency and mitigation plans (and due date and responsible parties for those plans).
- **Project Issue Log (updated)**, listing all project issues, response plans, due dates, and responsible parties for each issue.
- **Project Status Reports**, copies of all status reports delivered to the project client during the phase.

### Key Considerations

- All contract deliverables associated with this phase must be signed off by the client for the phase to be considered closed out.
- All internal deliverables associated with this phase must complete the phase review for the phase to be considered closed out.

### Task Guidance

#### **Publish Deliverable Statement(s) of Intent (DSOIs) to Client**

At the start of the project phase, prepare a **Deliverable Statement of Intent** for each deliverable to be produced and approved in the phase. A DSOI is a 1-2 page overview of the deliverable, and will typically define the deliverable format, contents, objectives and goals, reviewers, and acceptance criteria. The contents are often described in *table of contents* format, with a brief description of each section, which may be expanded or refined in the final deliverable.

#### **Prepare checklist of acceptance requirements for each deliverable**

The Project Manager, or designated project team member, will review the project contract to identify all processes, tasks, deliverables, and work products that are to be produced during this

phase of the project. All identified items are to be included in the checklist of acceptance requirements, for review during this activity.

**PQM 8.1 Prepare Production Build**

**Objective**

The objective of this activity is to prepare, package, and deliver a full build of the developed software for installation in the client’s Production environment.



**Phase**

This activity is performed in the Deployment phase.

DEPLOYMENT PHASE

Tasks	
PQM 8.1.1	Create a Production folder in the version control system
PQM 8.1.2	Copy the approved IV&V Build to the Production folder in the version control system
PQM 8.1.3	Update documentation recommended and/or permitted by the IV&V Report.
PQM 8.1.4	Prepare and package all baselined products for client delivery in accordance with the client’s Software Deliverable Guidelines. <ul style="list-style-type: none"> <li>• Update all documents to indicate the Production submission and date</li> <li>• Print all documents</li> <li>• Assemble all documents in a binder</li> <li>• Create all baselined software CDs according to the client’s standards</li> <li>• Create all baselined document CDs according to the client’s standards</li> </ul>
PQM 8.1.5	Verify Production software CD(s) – Test Installation Procedures from the Production software CD(s)
PQM 8.1.6	Prepare email notification of the Production build
PQM 8.1.7	Send email notification to the client Project Manager responsible for the release
PQM 8.1.8	Deliver Production build package to Project Manager for review and approval
PQM 8.1.9	Deliver Production build package to the client Project Manager, for Production installation

## Deliverable

The **Production Build** deliverable includes the documents, software CD(s), contact list, and any assistance offered to support the client's installation process for the project.

## Responsibilities

R – Quality Management Specialists

A – Quality Management Supervisor/Manager, Project Manager, Client Project Manager

S – Technical Team Lead, Application Designers/Developers

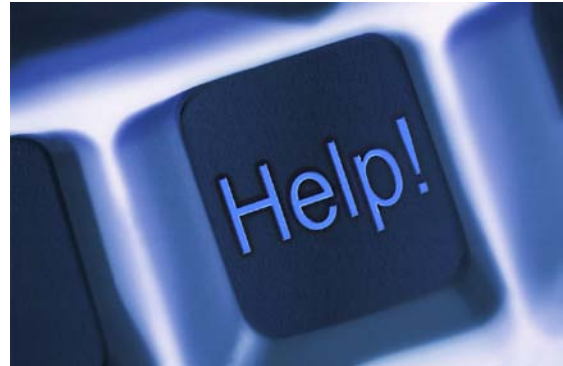
## Key Considerations

- The Project Manager should check with the Client Project Manager periodically to understand the work schedule of the Production installer(s). Support team members must be readily available to the Production installation team to respond to questions or problems, including after hours and weekends.
- The Production Build should include copies of all previous IV&V Reports and IV&V Response Reports in the build.

**PQM 8.2 Support Production Installation**

**Objective**

The client's Production installation team installs the Production Build with the support of the Development Team. The installation is validated and the client Project Manager is informed that the installation is ready for user access.



**Phase**

This activity is performed in the Deployment phase.

DEPLOYMENT PHASE

Tasks	
PQM 8.2.1	Verify that the Production Build has been received by the client
PQM 8.2.2	Verify that the Production Build has been assigned to the client's Production installation team
PQM 8.2.3	Monitor the build's status; Notify Development Team once installation procedures have begun
PQM 8.2.4	Verify that the Development Team is available to respond to questions or problems throughout the installation process, including after hours and weekends
PQM 8.2.5	Regularly communicate with the client Project Manager and the client's Production installation team to verify that Production installation activities are progressing
PQM 8.2.6	Monitor the build's status; notify Development Team once installation procedures are complete
PQM 8.2.7	The Development Team executes the Production Validation process to verify that the installation was properly executed
PQM 8.2.8	Transmit an email notification to the client Project Manager that the installation is complete, validated, and ready for user access

**Deliverable**

No deliverable is produced by this activity.

## Responsibilities

R – Technical Team Lead

A – Quality Management Supervisor/Manager, Project Manager, Client Project Manager

S – Quality Management Specialists, Application Designers/Developers

## Key Considerations

- The Project Manager should check with the client Project Manager periodically to understand the work schedule of the client's Production installer(s). Support team members must be readily available to the client's Production installation team to respond to questions or problems, including after hours and weekends.
- The Development Team should immediately acknowledge each email support request from the client's Production installation team during the installation processes. If additional time is needed, the Development Team should send a follow-up email with a course of action.

**PQM 8.9 Execute Deployment Phase Deliverable Review**

**Objective**

The objective of this activity is to conduct the processes that review the quality of the phase work products, verifying that the project deliverables meet the defined requirements and client expectations, achieve client acceptance, and close-out the project phase.



**Phase**

This activity is performed in the Deployment phase.

DEPLOYMENT PHASE

Tasks	
PQM 8.9.1	Prepare and Publish Deliverable Statement(s) of Intent (DSOIs) to Client
PQM 8.9.2	Review Phase DSOIs with Client
PQM 8.9.3	Document acceptance requirements for each deliverable
PQM 8.9.4	Document client and internal requirements for phase close-out
PQM 8.9.5	Schedule internal review(s) of phase deliverables
PQM 8.9.6	Schedule client review(s) of phase deliverables
PQM 8.9.7	Prepare checklist of acceptance requirements for each deliverable
PQM 8.9.8	Prepare checklist of phase close-out requirements
PQM 8.9.9	Inventory the phase deliverables
PQM 8.9.10	Conduct internal review(s) of phase deliverables <ul style="list-style-type: none"> <li>Review deliverable(s) against acceptance requirements</li> <li>Identify and document compliance issues, if any</li> <li>Identify and document remediation actions, if needed</li> <li>Schedule follow-up internal review, if needed</li> </ul>
PQM 8.9.11	Conduct client review(s) of phase deliverables <ul style="list-style-type: none"> <li>Review deliverable(s) against acceptance requirements</li> <li>Identify and document compliance issues, if any</li> <li>Identify and document remediation actions, if needed</li> <li>Schedule follow-up client review, if needed</li> <li>Document review results and publish to client</li> </ul>

Tasks	
PQM 8.9.12	Deliver phase deliverables to client for approval and sign-off
PQM 8.9.13	Collect client sign-off on phase deliverables
PQM 8.9.14	Schedule internal review of phase close-out requirements
PQM 8.9.15	Conduct internal review of phase close-out requirements <ul style="list-style-type: none"> <li>• Review status of phase deliverables against close-out requirements</li> <li>• Review status of remaining close-out requirements</li> <li>• Identify and document compliance issues, if any</li> <li>• Identify and document remediation actions, if needed</li> <li>• Schedule follow-up internal review, if needed</li> <li>• If ready, complete phase close-out</li> </ul>
PQM 8.9.16	Collect and document Lessons Learned

### Deliverable

A **Deliverable Statement of Intent** is prepared for each deliverable to be produced and approved in the phase. The DSOI describes the planned deliverable in sufficient detail that the client is able to evaluate the planned organization and content of the deliverable, and comment on the suitability of the planned deliverable. Achieving this agreement early in the phase is critical to deliverable acceptance later in the phase.

**Lessons Learned** are documented and collected at the end of each project phase, highlighting activities in need of adjustment or improvement to meet project objectives. Separate sessions are held for the internal project team, and the project team leadership and client participants of the project phase.

### Exit Criteria

Exit Criteria include certain deliverables which must be reviewed and approved (internally and with the client) before the project may proceed to the next phase. In addition to deliverables defined and established in the method(s) that will be engaged by project(s), as well as the method(s) used by the client and those specified in the project contract, the following deliverables must be completed, reviewed, and approved before this phase is completed:

- **Deliverable Schedule (final)**, listing all internal and contractual deliverables, for the project, and current due dates.
- **Project Decision Log (final)**, listing all project decisions made, including date of decision and all participants in the decision.
- **Project Risk Log (final)**, listing all identified project risks, including contingency and mitigation plans (and due date and responsible parties for those plans).

- **Project Issue Log (final)**, listing all project issues, response plans, due dates, and responsible parties for each issue.
- **Project Status Reports**, copies of all status reports delivered to the project client during the phase.

## Responsibilities

R – Project Manager

A – Project client

S – All project team members

I – Client contracting officer

## Key Considerations

- All contract deliverables associated with this phase must be signed off by the client for the phase to be considered closed out.
- All internal deliverables associated with this phase must complete the phase review for the phase to be considered closed out.

## Task Guidance

### **Publish Deliverable Statement(s) of Intent (DSOIs) to Client**

At the start of the project phase, prepare a **Deliverable Statement of Intent** for each deliverable to be produced and approved in the phase. A DSOI is a 1-2 page overview of the deliverable, and will typically define the deliverable format, contents, objectives and goals, reviewers, and acceptance criteria. The contents are often described in *table of contents* format, with a brief description of each section, which may be expanded or refined in the final deliverable.

### **Prepare checklist of acceptance requirements for each deliverable**

The Project Manager, or designated project team member, will review the project contract to identify all processes, tasks, deliverables, and work products that are to be produced during this phase of the project. All identified items are to be included in the checklist of acceptance requirements, for review during this activity.

## Appendix – Sections to be Added

The current version of this document is a work-in-progress. This first appendix is a list of outstanding sections to be added to the document in an upcoming version.

- Phase End Checklists
- Conduct Peer Code Reviews
- Unit Test Spot Check
- Generic IV&V Procedures
- Generic Deployment Procedures

## Appendix – Testing Glossary

**Accessibility Testing:** Verifying a product is accessible to the people having disabilities (deaf, blind, mentally disabled etc.). See *Section 508 compliance*.

**Ad Hoc Testing:** A testing phase where the tester tries to 'break' the system by randomly trying the system's functionality. Can include negative testing as well. See also *Monkey Testing*.

**Agile Testing:** Testing practice for projects using agile methodologies, treating development as the customer of testing and emphasizing a test-first design paradigm. See also *Test Driven Development*.

**Application Binary Interface (ABI):** A specification defining requirements for portability of applications in binary forms across different system platforms and environments.

**Application Programming Interface (API):** A formalized set of software calls and routines that can be referenced by an application program in order to access supporting system or network services.

**Automated Software Quality (ASQ):** The use of software tools, such as automated testing tools, to improve software quality.

### Automated Testing:

- Testing employing software tools which execute tests without manual intervention. Can be applied in GUI, performance, API, etc. testing.
- The use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.

**Backus-Naur Form:** A metalanguage used to formally describe the syntax of a language.

**Basic Block:** A sequence of one or more consecutive, executable statements containing no branches.

**Basis Path Testing:** A white box test case design technique that uses the algorithmic flow of the program to design tests.

**Basis Set:** The set of tests derived using *basis path testing*.

**Baseline:** The point at which some deliverable produced during the software engineering process is put under formal change control.

**Benchmark Testing:** Tests that use representative sets of programs and data designed to evaluate the performance of computer hardware and software in a given configuration.

**Beta Testing:** Testing of a rerelease of a software product conducted by customers.

**Binary Portability Testing:** Testing an executable application for portability across system platforms and environments, usually for conformation to an ABI specification.

**Black Box Testing:** Testing based on an analysis of the specification of a piece of software without reference to its internal workings. The goal is to test how well the component conforms to the published requirements for the component.

**Bottom Up Testing:** An approach to integration testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested.

**Boundary Testing:** Test which focus on the boundary or limit conditions of the software being tested. (Some of these tests are stress tests).

**Boundary Value Analysis:** In boundary value analysis, test cases are generated using the extremes of the input domain, e.g. maximum, minimum, just inside/outside boundaries, typical values, and error values. BVA is similar to Equivalence Partitioning but focuses on "corner cases".

**Branch Testing:** Testing in which all branches in the program source code are tested at least once.

**Breadth Testing:** A test suite that exercises the full functionality of a product but does not test features in detail.

**Bug:** A fault in a program which causes the program to perform in an unintended or unanticipated manner.

**CAST:** Computer Aided Software Testing.

**Capture/Replay Tool:** A test tool that records test input as it is sent to the software under test. The input cases stored can then be used to reproduce the test at a later time. Most commonly applied to GUI test tools.

**CMMI:** The Capability Maturity Model Integration for Software Development (CMMI-DEV) is a model for judging the maturity of the software processes of an organization and for identifying the key practices that are required to increase the maturity of these processes.

**Cause Effect Graph:** A graphical representation of inputs and the associated outputs effects which can be used to design test cases.

**Code Complete:** Phase of development where functionality is implemented in entirety; bug fixes are all that are left. All functions found in the Functional Specifications have been implemented.

**Code Coverage:** An analysis method that determines which parts of the software have been executed (covered) by the test case suite and which parts have not been executed and therefore may require additional attention.

**Code Inspection:** A formal testing technique where the programmer reviews source code with a group who ask questions analyzing the program logic, analyzing the code with respect to a checklist of historically common programming errors, and analyzing its compliance with coding standards.

**Code Review/Walkthrough:** A formal testing technique where source code is traced by a group with a small set of test cases, while the state of program variables is manually monitored, to analyze the programmer's logic and assumptions.

**Coding:** The generation of source code.

**Compatibility Testing:** Testing whether software is compatible with other elements of a system with which it should operate, e.g. browsers, Operating Systems, or hardware.

**Component:** A minimal software item for which a separate specification is available.

**Component Testing:** See *Unit Testing*.

**Concurrency Testing:** Multi-user testing geared towards determining the effects of accessing the same application code, module or database records. Identifies and measures the level of locking, deadlocking and use of single-threaded code and locking semaphores.

**Conformance Testing:** The process of testing that an implementation conforms to the specification on which it is based. Usually applied to testing conformance to a formal standard.

**Context Driven Testing:** The context-driven school of software testing is flavor of Agile Testing that advocates continuous and creative evaluation of testing opportunities in light of the potential information revealed and the value of that information to the organization right now.

**Conversion Testing:** Testing of programs or procedures used to convert data from existing systems for use in replacement systems.

**Cyclomatic Complexity:** A measure of the logical complexity of an algorithm, used in white-box testing.

**Data Dictionary:** A database that contains definitions of all data items defined during analysis.

**Data Flow Diagram:** A modeling notation that represents a functional decomposition of a system.

**Data Driven Testing:** Testing in which the action of a test case is parameterized by externally defined data values, maintained as a file or spreadsheet. A common technique in *Automated Testing*.

**Debugging:** The process of finding and removing the causes of software failures.

**Defect:** Nonconformance to requirements or functional / program specification

**Dependency Testing:** Examines an application's requirements for pre-existing software, initial states and configuration in order to maintain proper functionality.

**Depth Testing:** A test that exercises a feature of a product in full detail.

**Dynamic Testing:** Testing software through executing it. See also *Static Testing*.

**Emulator:** A device, computer program, or system that accepts the same inputs and produces the same outputs as a given system.

**Endurance Testing:** Checks for memory leaks or other problems that may occur with prolonged execution.

**End-to-End testing:** Testing a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

**Equivalence Class:** A portion of a component's input or output domains for which the component's behavior is assumed to be the same from the component's specification.

**Equivalence Partitioning:** A test case design technique for a component in which test cases are designed to execute representatives from equivalence classes.

**Error:** A mistake in the system under test; usually but not always a coding mistake on the part of the developer.

**Exhaustive Testing:** Testing which covers all combinations of input values and preconditions for an element of the software under test.

**Functional Decomposition:** A technique used during planning, analysis and design; creates a functional hierarchy for the software.

**Functional Specification:** A document that describes in detail the characteristics of the product with regard to its intended features.

**Functional Testing:** See also *Black Box Testing*.

- Testing the features and operational behavior of a product to ensure they correspond to its specifications.
- Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions.

**Glass Box Testing:** A synonym for *White Box Testing*.

**Gorilla Testing:** Testing one particular module, functionality heavily.

**Gray Box Testing:** A combination of *Black Box* and *White Box* testing methodologies: testing a piece of software against its specification but using some knowledge of its internal workings.

**High Order Tests:** Black-box tests conducted once the software has been integrated.

**Independent Test Group (ITG):** A group of people whose primary responsibility is software testing.

**Inspection:** A group review quality improvement process for written material. It consists of two aspects; product (document itself) improvement and process improvement (of both document production and inspection).

**Integration Testing:** Testing of combined parts of an application to determine if they function together correctly. Usually performed after unit and functional testing. This type of testing is especially relevant to client/server and distributed systems.

**Installation Testing:** Confirms that the application under test recovers from expected or unexpected events without loss of data or functionality. Events can include shortage of disk space, unexpected loss of communication, or power out conditions.

**Load Testing:** See *Performance Testing*.

**Localization Testing:** This term refers to making software specifically designed for a specific locality.

**Loop Testing:** A white box testing technique that exercises program loops.

**Metric:** A standard of measurement. Software metrics are the statistics describing the structure or content of a program. A metric should be a real objective measurement of something such as number of bugs per lines of code.

**Monkey Testing:** Testing a system or an Application on the fly, i.e just few tests here and there to ensure the system or an application does not crash out.

**Mutation Testing:** Testing done on the application where bugs are purposely added to it.

**Negative Testing:** Testing aimed at showing software does not work. Also known as "test to fail". See also *Positive Testing*.

**N+1 Testing:** A variation of Regression Testing. Testing conducted with multiple cycles in which errors found in test cycle N are resolved and the solution is retested in test cycle N+1. The cycles are typically repeated until the solution reaches a steady state and there are no errors. See also *Regression Testing*.

**Path Testing:** Testing in which all paths in the program source code are tested at least once.

**Performance Testing:** Testing conducted to evaluate the compliance of a system or component with specified performance requirements. Often this is performed using an automated test tool to simulate large number of users. Also know as "Load Testing".

**Positive Testing:** Testing aimed at showing software works. Also known as "test to pass". See also *Negative Testing*.

**Quality Assurance:** All those planned or systematic actions necessary to provide adequate confidence that a product or service is of the type and quality needed and expected by the customer.

**Quality Audit:** A systematic and independent examination to determine whether quality activities and related results comply with planned arrangements and whether these arrangements are implemented effectively and are suitable to achieve objectives.

**Quality Circle:** A group of individuals with related interests that meet at regular intervals to consider problems or other matters related to the quality of outputs of a process and to the correction of problems or to the improvement of quality.

**Quality Control:** The operational techniques and the activities used to fulfill and verify requirements of quality.

**Quality Management:** That aspect of the overall management function that determines and implements the quality policy.

**Quality Policy:** The overall intentions and direction of an organization as regards quality as formally expressed by top management.

**Quality System:** The organizational structure, responsibilities, procedures, processes, and resources for implementing quality management.

**Race Condition:** A cause of concurrency problems. Multiple accesses to a shared resource, at least one of which is a write, with no mechanism used by either to moderate simultaneous access.

**Ramp Testing:** Continuously raising an input signal until the system breaks down.

**Recovery Testing:** Confirms that the program recovers from expected or unexpected events without loss of data or functionality. Events can include shortage of disk space, unexpected loss of communication, or power out conditions.

**Regression Testing:** Retesting a previously tested program following modification to ensure that faults have not been introduced or uncovered as a result of the changes made.

**Release Candidate:** A pre-release version, which contains the desired functionality of the final version, but which needs to be tested for bugs (which ideally should be removed before the final version is released).

**Sanity Testing:** Brief test of major functional elements of a piece of software to determine if its basically operational. See also *Smoke Testing*.

**Scalability Testing:** Performance testing focused on ensuring the application under test gracefully handles increases in work load.

**Security Testing:** Testing which confirms that the program can restrict access to authorized personnel and that the authorized personnel can access the functions available to their security level.

**Smoke Testing:** A quick-and-dirty test that the major functions of a piece of software work. Originated in the hardware testing practice of turning on a new piece of hardware for the first time and considering it a success if it does not catch on fire.

**Soak Testing:** Running a system at high load for a prolonged period of time. For example, running several times more transactions in an entire day (or night) than would be expected in a busy day, to identify and performance problems that appear after a large number of transactions have been executed.

**Software Requirements Specification:** A deliverable that describes all data, functional and behavioral requirements, all constraints, and all validation requirements for software/

**Software Testing:** A set of activities conducted with the intent of finding errors in software.

**Static Analysis:** Analysis of a program carried out without executing the program.

**Static Analyzer:** A tool that carries out static analysis.

**Static Testing:** Analysis of a program carried out without executing the program.

**Storage Testing:** Testing that verifies the program under test stores data files in the correct directories and that it reserves sufficient space to prevent unexpected termination resulting from lack of space. This is external storage as opposed to internal storage.

**Stress Testing:** Testing conducted to evaluate a system or component at or beyond the limits of its specified requirements to determine the load under which it fails and how. Often this is *performance testing* using a very high level of simulated load.

**Structural Testing:** Testing based on an analysis of internal workings and structure of a piece of software. See also *White Box Testing*.

**System Testing:** Testing that attempts to discover defects that are properties of the entire system rather than of its individual components.

**Testability:** The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met.

**Testing:**

- The process of exercising software to verify that it satisfies specified requirements and to detect errors.

- The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs), and to evaluate the features of the software item (Ref. IEEE Std 829).
- The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component.

**Test Automation:** See *Automated Testing*.

**Test Bed:** An execution environment configured for testing. May consist of specific hardware, OS, network topology, configuration of the product under test, other application or system software, etc. The Test Plan for a project should enumerate the test beds(s) to be used.

**Test Case:**

- Test Case is a commonly used term for a specific test. This is usually the smallest unit of testing. A Test Case will consist of information such as requirements testing, test steps, verification steps, prerequisites, outputs, test environment, etc.
- A set of inputs, execution preconditions, and expected outcomes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

**Test Driven Development:** Testing methodology associated with Agile Programming in which every chunk of code is covered by unit tests, which must all pass all the time, in an effort to eliminate unit-level and regression bugs during development. Practitioners of TDD write a lot of tests, i.e. an equal number of lines of test code to the size of the production code.

**Test Driver:** A program or test tool used to execute a test. Also known as a Test Harness.

**Test Environment:** The hardware and software environment in which tests will be run, and any other software with which the software under test interacts when under test including stubs and test drivers.

**Test First Design:** Test-first design is one of the mandatory practices of Extreme Programming (XP). It requires that programmers do not write any production code until they have first written a unit test.

**Test Harness:** A program or test tool used to execute a test. Also known as a Test Driver.

**Test Plan:** A document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning. Ref IEEE Std 829.

**Test Procedure:** A document providing detailed instructions for the execution of one or more *test cases*.

**Test Scenario:** Definition of a set of *test cases* or *test scripts* and the sequence in which they are to be executed.

**Test Script:** Commonly used to refer to the instructions for a particular test that will be carried out by an automated test tool.

**Test Specification:** A document specifying the test approach for a software feature or combination of features and the inputs, predicted results and execution conditions for the associated tests.

**Test Suite:** A collection of tests used to validate the behavior of a product. The scope of a Test Suite varies from organization to organization. There may be several Test Suites for a particular product for example. In most cases however a Test Suite is a high level concept, grouping together hundreds or thousands of tests related by what they are intended to test.

**Test Tools:** Computer programs used in the testing of a system, a component of the system, or its documentation.

**Thread Testing:** A variation of *top-down testing* where the progressive integration of components follows the implementation of subsets of the requirements, as opposed to the integration of components by successively lower levels.

**Top Down Testing:** An approach to integration testing where the component at the top of the component hierarchy is tested first, with lower level components being simulated by stubs. Tested components are then used to test lower-level components. The process is repeated until the lowest level components have been tested.

**Total Quality Management:** A company commitment to develop a process that achieves high quality product and customer satisfaction.

**Traceability Matrix:** A document showing the relationship between Test Requirements and Test Cases.

**Usability Testing:** Testing the ease with which users can learn and use a product.

**Use Case:** The specification of tests that are conducted from the end-user perspective. Use cases tend to focus on operating software as an end-user would conduct their day-to-day activities.

**User Acceptance Testing:** A formal product evaluation performed by a customer as a condition of purchase.

**Unit Testing:** Testing of individual software components.

**Validation:** The process of evaluating software at the end of the software development process to ensure compliance with software requirements. The techniques for validation is testing, inspection and reviewing.

**Verification:** The process of determining whether or not the products of a given phase of the software development cycle meet the implementation steps and can be traced to the incoming objectives established during the previous phase. The techniques for verification are testing, inspection and reviewing.

**Volume Testing:** Testing which confirms that any values that may become large over time (such as accumulated counts, logs, and data files), can be accommodated by the program and will not cause the program to stop working or degrade its operation in any manner.

**Walkthrough:** A review of requirements, designs or code, characterized by the author of the material under review guiding the progression of the review.

**White Box Testing:** Testing based on an analysis of internal workings and structure of a piece of software. Includes techniques such as *Branch Testing* and *Path Testing*. Also known as *Structural Testing* and *Glass Box Testing*. Contrast with *Black Box Testing*.

**Workflow Testing:** Scripted end-to-end testing which duplicates specific workflows which are expected to be utilized by the end-user.

## Appendix – Sample ETL Unit Test Cases

### ETL Programs

The individual ETL Developer is responsible for conducting unit testing for his/her own ETL code. Unit testing the ETL programs is designed to identify and address issues with the implementation of the specific ETL modules. Typical tests include:

1. Configuration and runtime parameters operate correctly
2. I/O paths are correctly defined
3. Connections to databases work correctly
4. Lookup lists, joins, aggregations are efficient
5. All logical paths are exercised
6. Keys are correctly assigned
7. Referential integrity is enforced
8. Null values are correctly processed
9. Logs are produced
10. Business rules applied correctly
11. Data values are correctly mapped

### Run Schedule

The ETL Developer should test the run schedule. The validation of the run schedule is accomplished by setting up the ETL to run from the scheduler. There are several validation points. The scheduler must demonstrate the ability to detect and respond to the events that drive the initiation, completion or termination of the ETL process. The simplest event the scheduler needs to support is the occurrence of a moment in time. At a preset time, the scheduler runs an ETL process. Another common event that can trigger the ETL process is the arrival of a file. Finally, the completion of a process may trigger the initiation of the ETL process.

The scheduler must also demonstrate the ability to communicate with the ETL process and determine its completion status. In many cases, this is not communicated directly but via a defined protocol. In many cases, the test run will execute the ETL process by calling a script that runs the ETL job. The script itself may be invoked through a remote procedure call (RPC). The scheduler may detect the completion process of the shell script, but this may not be an accurate reflection of the completion status of the ETL job. In fact, the ETL job may complete successfully, but a detected condition may be interpreted as fatal. The intended completion status is buried in the ETL log. If this is written to a file, an agreed protocol for detecting, persisting and communicating status back to the scheduler will be defined. This test should validate that reporting mechanism.

Another test to validate the test run schedule is the validation of the parameters passed by the scheduler to the ETL process. These could include connection strings, root paths, criteria for limiting extracts etc. The unit test should validate that the agreed protocol for bundling and passing run time parameters works as expected.

In the end, the test run is successful if the right jobs run with the right parameters when the triggering condition occurs, and the intended status is communicated back to the scheduler. A test scenario should be defined that validates each execution path and validates the behavior of the scheduler and the ETL as run from the scheduler.

### Test / Validate Restart and Recovery Plan

Upon completion of unit testing of the ETL programs and their corresponding run schedule, the ETL Developer should validate the readiness of the restart and recovery plan. The intent of the restart and recovery validation is to thoroughly test the interfaces between the ETL modules and the job control infrastructure. This is accomplished by reproducing conditions under which each module would fail. For example, an input is missing, an output directory does not exist, duplicate primary key values in the load. The idea is to create the failure conditions and validate the ETL module can successfully report its status to the controlling layer.

A restart should bypass modules that have previously run and yield the same output set subsequent to a failure that would have been produced without a failure. An additional check should verify that if a module does run twice, it will generate the same end state as running it once. When designing the unit tests around restart-ability, particular attention should be paid to restart dependencies. For example, what are the transaction boundaries? If the ETL is set to commit every 50,000 records and the failure occurs in the middle of a transaction, did the database commit or rollback? Do we need to clean prior to restart? Does the job always perform an “upsert” making cleanup redundant? Are there record sort dependencies to enable restart? In every case the restart should produce the same end result as it would without failures.

The failure/restart/recovery test suite should validate the existing interfaces that were defined in the ETL architecture as well as validating the proper creation of the dataset while testing every logic path in the ETL module. This includes:

- 1) Persistence of completion state within an ETL cycle
- 2) Conditional execution based on current execution state (failed, pending, success)
- 3) Housekeeping required for clean restart
- 4) Logging of failure code, action taken, restart time
- 5) Notification of completion/failure status (to console, scheduler, operator)
- 6) Detection of restart conditions

### Test / Validate Initialization and Cleanup Plan

The ETL Developer should validate the initialization and cleanup logic of the ETL application. The initiation of the ETL cycle, or the termination of the ETL cycle, serves multiple purposes. From a restart perspective, the ETL modules are placed into a pending state or into a complete state. Work files are positioned; archiving activities are accomplished. The ETL architecture calls out the role of the initiate or terminate activities.

Typically, the initiation of the ETL cycle is marked by preparation of the working environment and a validation of the conditions required for a successful ETL run. Activities in the initiation sequence include:

- 1) Place the ETL modules within current cycle into a wait/pending state. This indicates that they are ready and waiting to be executed and they do not require a cleanup step.
- 2) Prepare any working directories or configuration files
- 3) Check prerequisites, like availability of source files, completion of processes, time windows etc.
- 4) Archive any previous ETL files that need to persist

The termination sequence for the ETL cycle may include:

- 1) Cleanup of transient files
- 2) Logging the completion status of the ETL cycle
- 3) Notification of cycle completion to scheduler, console, operator.
- 4) Collection and publication of performance and runtime metrics
- 5) Collection and reporting of data quality metrics (RI report, recycle records etc.)

### Test / Validate Purge and Archive Plan

In general, the data in the ETL landing, staging, and warehouse areas has a finite retention schedule. The classes of retention can vary from as short as the life of the ETL module, the cycle or they may persist for some longer period of time. In general, the files, tables or rows that persist beyond the life of the ETL module have a planned obsolescence. The aging process typically moves the data from the working environment to an archive and eventually it is purged. The aging process may be accomplished as an integrated portion of the ETL cycle or may run asynchronously between cycles. In either case, the testing of the plan involves validating that files, tables or data rows are moved according to the plan defined in the warehouse architecture.

In order to test the purging of files or tables that are intended to persist to the end of the ETL cycle:

- 1) Define criteria for identifying the files. This may be an inventory list, populated by the process that builds the file. It may be all transient files are assigned to a specific directory. In any case, a defined process for acquiring the set needs to be defined.
- 2) Run the script that is intended to purge the files and ensure they are deleted.
- 3) In the case of tables, the same methodology of preparing an inventory is identified. Temporary tables should always be segregated into a special schema.
- 4) For tables, decide whether the table is to be purged or the contents of the table.
- 5) Ensure the process that purges temporary tables does not have rights to drop permanent tables.

The second class of items that will get purged or archived have a retention schedule based on either time or some other triggering condition. For example, recycled records may persist for a specified number of days or a specified number of execution cycles. In either case, these class of objects typically age to a point where they can be removed from the system. This includes:

- 1) Moving the set of items to a defined archive directory, possibly compressing as part of the process.
- 2) Test the ability of the archiving script to identify the set of files to be archived.
- 3) Test any prerequisites to archiving, like completion status of the ETL cycle.

- 4) Test the ability to detect age or retention status of previously archived objects.
- 5) Test the ability to re-introduce archived objects back into the working environment.

A final class of purge processing that needs to be tested is row purging from the warehouse tables. Purging rows from the warehouse is typically based on age. A careful analysis of the purge strategy must be completed to ensure the proper alignment of fact purging. For example, an aggregate that is dependent on atomic records will need to be recomputed or if there is a perfect alignment, purge the aggregate records. Data purging is not only for purposes of space reclamation, but also to limit legal exposure beyond required retention schedules. The process is similar to file archiving:

- 1) Validate the ability to identify candidate rows for purging
- 2) Validate ability to identify dependent tables/rows
- 3) Validate permissions to drop rows
- 4) Validate strategy to delete rows or drop partitions
- 5) Validate authorization process – are rows marked for deletion and then approved

### Test / Validate Rejected Record Processing Procedures

The ETL Developer should validate the proper handling of the rejected records created from the ETL run stream. Records may be rejected for any number of reasons, however the causes can be roughly grouped into the violation of business or physical constraints. On the business side, the rules are implemented in the ETL processing logic. On the physical side the rejection occurs because of an unexpected condition. For example, on the business side we might reject a row with a date that is in the future if the business rule states that dates must be in the past and this should trigger a reject. A row may be rejected by the system if a column violates a database constraint.

The validation suite for testing reject processing should first determine that rows are in fact rejected in accordance with the intent of the business rules and rows rejected due to physical constraints are not lost. In both cases, a test data set should be prepared and run through the ETL module. The output rows must be accounted for in order to balance the input and output sides of the process. A test of reject processing should include:

- 1) Identification of all business rules that trigger a reject
- 2) Preparation of a dataset that does, and does not, violate each rule
- 3) Run the data set through the ETL process
- 4) Validate each condition routes the record to the appropriate destination.
- 5) Validate the reported counts of input records, output records and reject records can be balanced.
- 6) Validate the disposition of the rejected records.

The ability to reject records intentionally or capture unexpected rejects is only half the validation task. The second aspect of validating the reject processing is the handling of rejected records. The strategy for handling business or physical reject records is defined as part of the ETL architecture. Typically, this will include a process for either fixing the record at the source system or isolating/tagging the record for attention prior to loading it to the warehouse. In either

case there is likely an aspect of notification, stewardship and reintroduction of the record in to the ETL process. Validation of these steps includes:

- 1) Validate the notification infrastructure. How are record counts collected and distributed
- 2) Validate the disposition of reject records. Are records rejected for business rule violations handled in the same way as records rejected for violation of physical constraints.
- 3) Validate the methodology and process of editing the rejected records.
- 4) Validate the re-introduction of the rejected record into the ETL process. What happens when a reject record is re-introduced and received from the source at the same time.
- 5) Validate the ability and willingness of stewards to determine the disposition of rejected records.

### Test / Validate Error Handling Procedures

The ETL Developer should validate the error handling procedures. ETL errors can be introduced either at the process level or at the data level. At the process level the severity of the error may require the ETL process to stop, address the problem and restart. Likewise, detected errors at the data level may trigger a similar decision. For example, at the process level and script may finish with warnings. At the data level row counts may exceed a pre-defined limit, or an out of balance condition may be detected. In either case the ETL architecture should have a defined framework for error logging, handling and notification.

Testing the process involves:

- 1) Creating the conditions where a process error would occur. Example scenarios include network outage, database down, system restarts etc. These are unexpected conditions outside the control of the ETL process. Can these be detected, logged and handled.
- 2) Create conditions that can be detected and validate the detection, logging and handling of the error. Examples include disk volume is full, expected input file is not present.
- 3) Validate the error handling of the ETL process to the error conditions. Does the ETL process stop when expected, attempt to fix the error condition or ignore it. Are concurrent processes notified.
- 4) Validate the notification process to the scheduler, console, operator. Is there an attempt to route classes of errors to a specific group. For example validate whether DBA gets notified if detected error is database related.

### Validate Capacity Plan

The ETL Developer, with the support of the ETL Designer and ETL Architect, should validate the adequacy of the processing capacity of the ETL application. The validation of the capacity plan is accomplished by comparing actual resource consumption to predicted requirements. This is accomplished by comparing the actual required storage, time and memory to the predicted. With an understanding of the available time for processing can the actual volume of data be processed in the allocated time. Measuring time constraints includes:

- 1) Measure the elapsed time of the ETL module with a known volume of data. Are time requirements directly (linear) proportional to the volume or does time per record increase with increased volume.
- 2) What is the overhead in running multiple modules together.

Disk space is often a concern due to the need for both permanent space requirements and transient requirements during processing.

- 1) For each ETL module determine the high-water mark for disk storage as a percentage of the total volume.
- 2) Run a known volume of data and measure the actual storage requirements in the database and at each stage of processing. Can the space be reclaimed before the end of the ETL cycle?
- 3) Based on temp space requirements how many ETL modules can run concurrently.
- 4) Does the measured space requirement validate the original estimate or require an adjustment to the estimation model.

Insufficient system capacity can have an adverse effect on the ETL module if the memory requirements exceed what is available. Likewise, exceeding available CPU cycles can result in degraded performance. For each ETL module:

- 1) Measure CPU utilization while the module is running. Validate the number of concurrent processes that can run while not over utilizing the available CPU.
- 2) Measure available memory to ensure excessive swapping is not occurring
- 3) Measure IO capacity and utilization of ETL module.

## Appendix – Kaizen Events

Kaizen, or rapid improvement processes, often is considered to be the "building block" of all lean production methods. Kaizen focuses on eliminating waste, improving productivity, and achieving sustained continual improvement in targeted activities and processes of an organization.

During a Kaizen Event, a small cross-functional team is assigned specific goals to achieve within a set period of time, typically one to five days. Compared to traditional part-time teams, Kaizen Teams typically spend half as many man-hours and complete their projects in 1/20th of the lead-time. Also, Kaizen Teams develop more creative and effective solutions.



The best knowledge resides with the people who actually perform the work. They know the problems and often the solutions. During a Kaizen Event, they make the recommendations on how to improve the process and they make the physical changes to the processes. They will also support and continue the process after the event is over.

Because the people who have to live with the processes on a daily basis are the people who study the current process, design the improved process, and then physically make the changes to convert to the new process, there is tremendous involvement, buy-in and ownership of the improvements. The changes created through the Kaizen event are very sustainable. The processes do not revert back to the less efficient way of doing things.

The Kaizen Event Process:

- Prior to the event, management creates an Event Mandate which establishes goals, boundaries, team members and schedule.
- During the event, the team reviews applicable techniques, analyzes the existing situation (including gathered evidence, such as Lessons Learned), develops and selects appropriate changes.
- Based on the above review, the selected changes are implemented immediately and resources are subsequently assigned to any open action items.
- At the end of each day, the team presents to management what has been accomplished and what remains to be done. Management provides support, feedback and any help the team needs to achieve the goals in the mandate.

Using this process, a dedicated team will typically accomplish more over several days than part-time individuals or teams will accomplish over several months or years. Also, the skills and

teamwork developed during this period will help create a team-oriented, continuous improvement environment after the event.

One of the key concepts of Kaizen is that “If there is No Action there can be No Success.” The goal is not a 100% solution that solves all the problems at one time, but rather a 60% solution that can be accomplished in a one-week time frame with the intent to hold another event in several months to further improve the processes.

A Kaizen Event is not a license to spend and should be accomplished with very little expenditure. Overall emphasis is placed on creating solutions and improvements with existing assets. As a result, they are a very cost-effective method to create dramatic improvements in processes.

A typical Kaizen event is one week long. A team is usually a cross-functional team that is composed of approximately 8 people. The team is composed of people who are in the process to be reviewed, such as the process workers, lead people, and supervisor. Additional resources from other departments are assigned to support the event. Personnel from Project Management, Development Engineering, Quality Management, Purchasing, Help Desk, Technical Support, Training, Finance, Business Development, etc., can be used to add value to the result of the event.

Training is done the first and second morning in the classroom. In the afternoon, the tools that were taught are applied by gathering data. Metrics for the current “as-is” process are established during the first afternoon. A report is made each afternoon to the group and other teams to exchange ideas. On Wednesday afternoon, the team should have a proposal put together as to what changes are proposed. The proposal includes the new metrics, proposed process flow, process map, process flow analysis, and spaghetti diagrams. Once the proposal is approved, the team can then start implementing the changes. The scope of the plan is to be able to complete the changes and have the new improved process up and running by Monday. So when the employees come in on Monday, the new processes are in place.

### Typical One-Week Kaizen Event Plan

<b>Day 1 - AM</b>	<b>Day 2 - AM</b>	<b>Day 3 - AM</b>	<b>Day 4 - AM</b>	<b>Day 5 - AM</b>
Training:	Training:	Floor Work:	Floor Work:	Floor Work:
<ul style="list-style-type: none"> <li>• Introduction</li> <li>• Kaizen Key Concepts</li> <li>• Process Flow Analysis</li> <li>• Spaghetti Diagrams</li> </ul>	<ul style="list-style-type: none"> <li>• Block Exercise</li> <li>• Takt Time &amp; Cycle Time</li> <li>• Pull Systems</li> <li>• Mistake Proofing</li> </ul>	<ul style="list-style-type: none"> <li>• Develop Detail Improvement Plan</li> <li>• Set Goals</li> </ul>	<ul style="list-style-type: none"> <li>• Implement Improvements</li> </ul>	<ul style="list-style-type: none"> <li>• Complete Improvements</li> <li>• Prepare Presentation to Management</li> </ul>

Day 1 - PM	Day 2 - PM	Day 3 - PM	Day 4 - PM	Day 5 - PM
Floor Work:	Floor Work:	Floor Work:	Floor Work:	Floor Work:
<ul style="list-style-type: none"> <li>• Process Mapping</li> <li>• Flow Analysis</li> <li>• Spaghetti Diagrams</li> <li>• Team Reports</li> </ul>	<ul style="list-style-type: none"> <li>• Process Observations</li> <li>• Takt-Cycle Time Chart</li> <li>• Proposed Layout</li> <li>• Team Reports</li> </ul>	<ul style="list-style-type: none"> <li>• Approve Plan</li> <li>• Start to Implement Improvements</li> <li>• Team Reports</li> </ul>	<ul style="list-style-type: none"> <li>• Implement Improvements</li> <li>• Team Reports</li> </ul>	<ul style="list-style-type: none"> <li>• Presentation to Management With Specific Measurables</li> </ul>

The **Spaghetti Diagram** is named so because the lines movement drawn on this diagram come to resemble a pile of tangled noodles. It is a movement path diagram, known by a more appetizing name.

**Takt time** requires one to consider an entire enterprise when designing a process, while cycle time would have one concentrate on just the task at hand, whatever it takes to make a unit in as short a time as possible. This comparison can be extrapolated a bit using concepts from physics. Even though cycle time has the word cycle in it, it typically focuses only on the "up" side of the cycle, where the item is being made. It essentially ignores the "down" side of the cycle, which can be idle time, the preparation for the next cycle to take place. Takt time would consider both the "up" and "down" sides of the production cycle. What this does is provide the practitioner with an expectation as to how the process should behave. It brings predictability into the design effort.

A great deal has been written, and continues to be written, about Kaizen, Kaizen Events, and Lean Six Sigma. Suggested readings include:

[The Kaizen Philosophy](#)

[Kaizan Rapid Process- epa.org](#)

[Kaizen Events and Project Management \(A Synergy\) – pmi.org](#)

[Personal and Professional Improvement with Kaizen](#)

[Developing A Kaizen Scorecard](#)

[Understanding Kaizen Metrics](#)

[Efficient Project Management: Combining Kaizen Philosophy with Lean and Six Sigma](#)

[An Analysis Of Lean Six Sigma](#)

[Do Kaizen Events help in Non-manufacturing units?](#)

## Appendix – Quality Management Plan Template

1. Overview
  - 1.1 Introduction
  - 1.2 Project Objectives
  - 1.3 Quality Management Objectives
  
2. <Project Name> Project
  - 2.1 Overview
  - 2.2 Project Scope
  - 2.3 Project Assumptions
  - 2.4 Project Organization
  - 2.5 Project Process
  - 2.6 Project Timeline
  - 2.7 Project Scope Management
  
3. Project Quality Management
  - 3.1 Overview
  - 3.2 Project Quality Management Organization
  - 3.3 Project Quality Management Process
  - 3.4 Quality Management Deliverables
  - 3.5 Quality Monitoring Metrics
  - 3.6 Problem Tracking, Reporting, and Corrective Actions
  - 3.7 Testing Methodology

## Appendix – Sample Quality Management Plan

### 1. Overview

#### 1.1 Introduction

The Data Sharing Committee (DSC) is a fictional committee of the U.S. Senate, established in 1976 to promote data and information sharing between civil federal agencies. The DSC sponsors projects that facilitate easy access to authoritative data managed by civil agencies.

#### 1.2 Project Objectives

The RIO project will integrate several Internet proposed technologies to support ad-hoc data sharing. The objective is to support the programmatic discovery and sharing of data with minimal human intervention.

Semantic web technologies, such as Resource Description Framework (RDF) and Web Ontology Language (OWL), will be combined with ontology development practices and traditional data management technologies, such as commercial database management and data integration products.

#### 1.3 Quality Management Objectives

The project's Quality Management efforts will verify that every deliverable fulfills the objectives, standards, and metrics established for that deliverable. Deliverables will be constructed and presented in a manner that supports timely review. The tests of the working system will demonstrate the ability to process real-world transactions and achieve the project's success criteria.

### 2. RIO Project

#### 2.1 Overview

The H1N1 crisis, international terrorism, and hundreds of cataclysms that occur across our globe each year... These and other far-reaching crises and threats know no borders.

To meet the demands of an increasingly interconnected world, more and more organizations – from the Department of Homeland Security to the Centers for Disease Control – are facing the challenges of sharing information, efficiently and in a timely manner.

These challenges include effectively assembling and organizing data to produce actionable intelligence, supporting clients' access requirements, employing effective

information publication techniques, managing security, and complying with privacy legislation.

## 2.2 Project Scope

The RIO project team will work with Senate data experts to define data classification rules (taxonomy) for the Senate Shared Data Set (SSDS), a collection of organizational and contact data for the committees and membership of the U.S. Senate. The project team will:

- develop a common vocabulary (ontology) for the in-scope data categories;
- analyze and align data quality with Senate, and related party, standards and business rules;
- define protocols for establishing automated data sharing partnerships; and
- integrate COTS technologies to organize and manage content and deliver secure access, timely publication, and privacy law compliance.

## 2.3 Project Assumptions

The first RIO project will focus on two primary data sets: U.S. Senate organizational structure, including committees, and membership of the U.S. Senate, including senators and support staff.

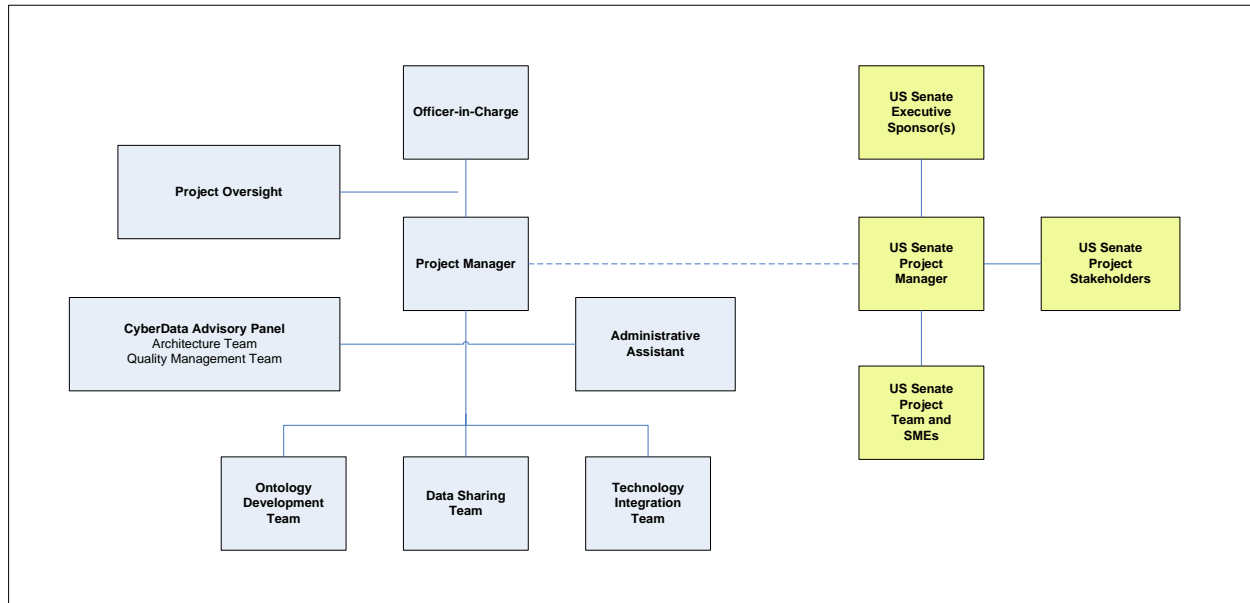
Availability of and commitment from U.S. Senate Subject Matter Experts (SMEs), stakeholders, and other support staff is assumed. This project is critically dependent on the active participation of these Senate personnel. Insufficient contribution of time and expertise by these staff members will adversely impact the project.

## 2.4 Project Organization

The figure below shows CyberData's project team organization for the RIO project, depicting the relationships among the CyberData team members, and the interfaces between the CyberData and U.S. Senate teams.

The project organizational structure supports a team philosophy that seeks collaboration between the U.S. Senate and CyberData, where team members communicate along appropriate organizational lines and the required technical and business interfaces are accounted for.

### Exhibit 1 – RIO Project Team Organization



The following table describes the roles and responsibilities by functional area for both U.S. Senate and CyberData.

**Exhibit 2 – Roles and Responsibilities by Functional Area**

Project Organization	U.S. Senate	CyberData
Project Oversight	<ul style="list-style-type: none"> <li>• Provide top-level direction</li> <li>• Review project status</li> <li>• Formally accept deliverables</li> <li>• Discuss concerns/issues as they are identified</li> </ul>	<ul style="list-style-type: none"> <li>• Provide corporate oversight and control</li> <li>• Approve major deliverables for delivery to the U.S. Senate</li> <li>• Make decisions on financial, contractual, and staffing matters</li> <li>• Ensure QA/QC procedures are followed</li> <li>• Review plans for resolving major issues</li> <li>• Receive feedback from U.S. Senate Oversight and U.S. Senate Project Management</li> </ul>
Project Management	<ul style="list-style-type: none"> <li>• Provide day-to-day supervision of project</li> <li>• Coordinate participation of U.S. Senate units in project</li> <li>• Review &amp; track project status</li> <li>• Work with CyberData Project Manager</li> </ul>	<ul style="list-style-type: none"> <li>• Provide day-to-day management of the project team</li> <li>• Set functional and technical direction</li> <li>• Establish schedules</li> <li>• Make staff assignments</li> <li>• Manage interaction with U.S. Senate</li> <li>• Review and approve all deliverables</li> </ul>

Project Organization	U.S. Senate	CyberData
	<ul style="list-style-type: none"> <li>• Review and comment on deliverables</li> <li>• Make decisions regarding project scope</li> <li>• Discuss issues and concerns</li> <li>• Respond to support requests</li> <li>• Maintain current understanding of project status</li> </ul>	<ul style="list-style-type: none"> <li>• Maintain frequent contact</li> <li>• Notify of needed U.S. Senate staff resources</li> <li>• Provide regular project status reports</li> <li>• Plan for team resource allocation</li> <li>• Address concerns about deliverables</li> <li>• Identify potential scope issues</li> </ul>
Advisory Panel	<ul style="list-style-type: none"> <li>• N/A</li> </ul>	<ul style="list-style-type: none"> <li>• Provide management and technical advice to the project</li> <li>• Assess project status and progress</li> </ul>
Architecture Team	<ul style="list-style-type: none"> <li>• N/A</li> </ul>	<ul style="list-style-type: none"> <li>• Review current U.S. Senate enterprise architecture and strategic IT goals</li> <li>• Review available technologies and tools</li> <li>• Plan and recommend alternative architecture solutions</li> </ul>
Quality Management Team	<ul style="list-style-type: none"> <li>• Review deliverables for quality and compliance</li> </ul>	<ul style="list-style-type: none"> <li>• Review deliverables for quality and compliance</li> </ul>
Ontology Development Team	<ul style="list-style-type: none"> <li>• N/A</li> </ul>	<ul style="list-style-type: none"> <li>• Conduct interviews with selected users and stakeholders, and assist in coordinating other U.S. Senate participation</li> <li>• Prepare requirements, business use case, and data analysis documentation</li> <li>• Work with technical team to translate business requirements to technical requirements</li> <li>• Develop vocabulary, including ontology models, for the in-scope data categories</li> <li>• Capture and document data quality rules and standards</li> </ul>
Data Sharing Team	<ul style="list-style-type: none"> <li>• N/A</li> </ul>	<ul style="list-style-type: none"> <li>• Meet with representative federal agencies' data sharing teams</li> <li>• Identify data sharing technical use cases</li> <li>• Define protocols for establishing automated data sharing partnerships</li> <li>• Develop RDF grammar</li> <li>• Develop system design and specifications</li> <li>• Validate design, business rules, and data</li> <li>• Link requirement to design component</li> </ul>

Project Organization	U.S. Senate	CyberData
		<ul style="list-style-type: none"> <li>• Maintain CASE tools</li> <li>• Conduct quality assurance and testing reviews</li> <li>• Provide documentation support</li> <li>• Map RDF/OWL to database structures</li> <li>• Develop/unit test software</li> <li>• Create prototypes</li> <li>• Program application modules</li> <li>• Create system interfaces</li> <li>• Integrate system modules</li> <li>• Test the system</li> </ul>
Technology Integration Team	<ul style="list-style-type: none"> <li>• N/A</li> </ul>	<ul style="list-style-type: none"> <li>• Integrate COTS technologies to organize and manage content and deliver secure access, timely publication, and privacy law compliance</li> <li>• Provide HW/SW installation procedures</li> <li>• Administer development environment</li> <li>• Implement physical database</li> </ul>

## 2.5 Project Process and Timeline

The figure below depicts CyberData’s method for executing the project services in the scope of the RIO project.

Exhibit 3 – RIO Project Plan



## 2.6 Project Scope Management

One of the most critical project management practices, scope management describes how agreed-upon changes will be incorporated into the project deliverables.

Scope Management is not the same as allowing Scope Creep. Change is a normal process in managing an organization. Scope Creep occurs when changes are added to the project, with little or no communication or consideration of impact to the project’s

deliverables, schedule, or cost. Scope Management allows projects to manage the adoption of change, while also managing expectations.

Before a project begins, the scope of the project is documented in a Scope Statement. The Scope Statement identifies the project objectives and deliverables, providing the basis for confirming understanding of project scope among the stakeholders.

The Scope Statement should include the following information:

1. **Strategy** -- An overview of the customer's business needs in relation to the project and the business need the project was undertaken to address.
2. **Product of the Project** – A summary of the project deliverables.
3. **Project Objectives** – Quantifiable goals in terms of time, money and technical quality that the project must achieve to be considered successful.
4. **Supporting Detail** – Description of all assumptions and constraints considered during the development of the scope statement.
5. **Scope Management Plan** – A description of how the project scope will be managed and how agreed changes will be incorporated into the project deliverables.

The need for scope change can come from:

- Stakeholders
- Project Team Members
- The environment
- Product obsolescence
- Technological advances
- Funding changes

As scope changes are identified, the Project Manager follows a specific process to bring the scope change to resolution:

1. The scope change is documented, usually in the form of a "Change Request" as well as appearing as an entry in the project's Change Log
2. The project team documents a description of the change, its source, justification and/or benefits, as well as impact to deliverables, project schedule, and project cost
3. The project team analyzes the Change Order to identify potential alternatives
4. On a regular basis, the Project Manager meets with the project's Change Control Board, where change requests are presented for resolution
5. The Change Control Board determines which change requests to approve, reject, or table (often requesting additional information)

6. Approved changes are communicated to the project stakeholders
7. Project baselines, used for performance monitoring, are updated
8. The project scope statement is updated
9. The project plan and budget are updated
10. The change is implemented and documented

By giving the project stakeholders a process for managing change, they are in a better position to assess the impact of those changes and can make informed decisions for approval or rejection of the change.

### **3. Project Quality Management**

#### **3.1 Overview**

Projects rarely get the opportunity to redo a poor job – the work must be done right the first time, or resources, efforts, and opportunities are wasted. Quality management establishes a quality plan that describes how the project will meet its quality objectives; quality assurance evaluates project performance on a regular basis to provide confidence that the project will satisfy the relevant quality objectives; and quality control monitors specific project results to determine if they comply with relevant quality standards.

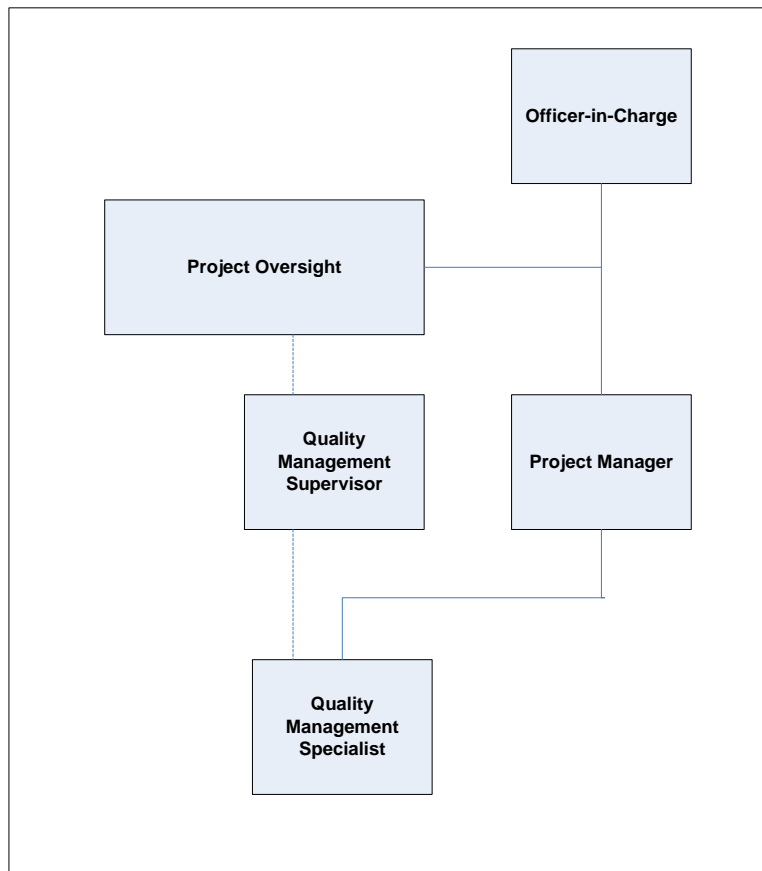
Delivery of quality client services is of singular importance to our team. Extensive quality management plans and guidelines are in place to provide customer satisfaction. CyberData's Project Quality Management (PQM), based on Total Quality Management (TQM) and Lean Six Sigma practices, integrates quality assurance and quality control procedures in each phase of a project. Each phase defines the expectations for the phase deliverables, verifies that the phase deliverables meet those expectations, and confirms that the project deliverables reasonably describe what needs to be done in the remaining phases of the project to meet client objectives.

This method provides detailed guidelines and forms to verify that time tested and proven management procedures are used to deliver high quality products.

#### **3.2 Project Quality Management Organization**

The figure below shows CyberData's Quality Management team organization for the RIO project, depicting the relationships among the CyberData team members.

#### **Exhibit 4 – Project Quality Management Organization**



### 3.3 Project Quality Management Process

The Project Quality Management method is divided into project phases, activities, tasks, work products, and deliverables:

- **Project Phase** – A grouping of activities that produce a set of deliverables that will support the project phase that follows.
- **Activity** – A collection of related tasks that produce a deliverable.
- **Task** – A discrete work unit that contributes to the production of a deliverable, including producing a component of a deliverable. For example, some tasks produce individual sections of a document deliverable.
- **Work Product** – A tangible, physical result of a project activity or task.
- **Deliverable** – A work product that is named in the project contract, often described with specific contents and delivery date

#### Exhibit 5 – Quality Management Activities by Project Phase

DEFINITION	REQUIREMENTS	HIGH-LEVEL DESIGN	LOW-LEVEL DESIGN	CONSTRUCTION	TESTING	IV&V	DEPLOYMENT	USE
PQM 1.1 Define Quality Management Plan	PQM 2.1 Prepare System Test Plan	PQM 3.1 Prepare Quality Architecture Design	PQM 4.1 Define Functional Unit Test Cases	PQM 5.1 Educate IV&V Team	PQM 6.1 Execute Unit Testing	PQM 7.1 Prepare IV&V Build	PQM 8.1 Prepare Production Build	
		PQM 3.2 Prepare Testing Environment Design	PQM 4.2 Define Integration Test Cases	PQM 5.2 Define Test Scripts	PQM 6.2 Build Release Candidate	PQM 7.2 Educate Production Team	PQM 8.2 Support Production Installation	
		PQM 3.3 Review Project with IV&V Team	PQM 4.3 Define System Test Cases	PQM 5.3 Define Automated Test Scripts	PQM 6.3 Update IV&V Team	PQM 7.3 Support IV&V Testing		
			PQM 4.4 Define User Acceptance Test Cases		PQM 6.4 Execute Integration Testing	PQM 7.4 Review and Respond to IV&V Report		
			PQM 4.5 Define Performance Test Cases		PQM 6.5 Execute System Testing			
			PQM 4.6 Install and Configure Testing Environment		PQM 6.6 Execute User Acceptance Testing			
					PQM 6.7 Execute Performance Testing			
PQM 1.9 Execute Definition Phase Deliverable Review	PQM 2.9 Execute Requirements Phase Deliverable Review	PQM 3.9 Execute High-Level Design Phase Deliverable Review	PQM 4.9 Execute Low-Level Design Phase Deliverable Review	PQM 5.9 Execute Construction Phase Deliverable Review	PQM 6.9 Execute Testing Phase Deliverable Review	PQM 7.9 Execute IV&V Phase Deliverable Review	PQM 8.9 Execute Deployment Phase Deliverable Review	

CyberData will facilitate project reviews with the Senate RIO Project Manager and Subject Matter Experts (SMEs). In addition, the project team will conduct reviews throughout the project to ensure that its activities are in accordance with the project plan and that the results meet the Senate RIO Project Manager’s expectations. Types of reviews that will be conducted include:

- Stakeholder and Functional Reviews** – Stakeholder and Functional Reviews will be performed throughout the project’s Requirements activities. All meeting minutes, requirements descriptions, and other user-oriented documents and results will be reviewed with stakeholders and project leadership.
- Advisory Committee Reviews** – The project team will engage IT architecture and quality management subject matter experts to verify that work products and deliverables support industry standards and meet CyberData’s and client’s performance expectations.
- Technical Reviews** – The RIO project team has ultimate responsibility for the operating performance of the delivered system and will conduct the technical reviews to assess the effectiveness and performance of the system at the design and development stages. The project team leadership will participate in these technical peer reviews.
- Management Report Reviews** – Management reports, project status, and measurement data will be analyzed to identify trends, potential quality issues,

process inefficiencies, and candidate improvement areas. Regular status reports will include review findings and results, as well as planned activities.

CyberData's Project Quality Management method executes quality control and assurance processes in every phase of the project process. Before each project deliverable is ready for client review, CyberData will evaluate it to make sure that it meets or exceeds client expectations and is prepared with the lowest possible chance of defects.

To ensure that the project process is accurately followed, the Quality Assurance team will perform regular internal audits. During these audits, QA will check the project organization, documentation, and quality records against client and CyberData guidelines and metrics, as well as general industry guidelines and best practices.

CyberData uses written test plans and protocols (test cases) to perform the Testing phase of each project. QA personnel are proficient in all types of testing: functional, user interface, white-box, black-box, regression, stress, performance, configuration, compatibility, and integration. The CyberData testing process includes proposed automation tools like Mercury Test Suite for unit, integration, end-to-end, regression, load, and stress testing.

### 3.4 Quality Management Deliverables

At the start of each project phase, CyberData will prepare a **Deliverable Statement of Intent** for each deliverable to be provided to the U.S. Senate. The DSOIs produced by the RIO project team will define deliverable format, contents, objectives and goals, reviewers, and acceptance criteria. The contents will be described in *table of contents* format, with a brief description of each section, which may be expanded or refined in the final deliverable.

Quality Management participates in the review and completion of all project deliverables, including communications, such as status reporting discussed in the previous section. The following table summarizes the major Quality Management deliverables to be produced by the project:

Exhibit 6 – Quality Management Deliverables by Project Phase

Project Phase	Quality Management Deliverable
Definition	Quality Management Plan
Requirements	System Test Plan
High-Level / Architecture Design	Quality Architecture Design
	Testing Environment Design
Low-Level / Application Program Design	Functional Unit Test Cases
	Integration Test Cases
	System Test Cases
	Performance Test Cases
	User Acceptance Test Cases
	Updated Requirements Traceability Matrix
	Testing Environment
Development / Construction	IV&V Presentation
	Test Scripts
	Automated Test Scripts
Testing	Build Plan
	Release Candidate
	IV&V Final Presentation
	Integration Testing Results
	System Testing Results
	Performance Testing Results
	User Acceptance Testing Results
IV&V	Production Team Presentation

### 3.5 Quality Monitoring Metrics

Performance metrics are fundamental to accurately tracking and effectively measuring the progress and quality of the project. The metrics are established early in the planning process and are easily measurable as the task progresses. The exhibit below is an example of a set of performance metrics to support the project’s development and implementation effort. This exhibit includes the performance metric, measurement approach, and parameters for successful execution. Actual performance metrics are identified, with stakeholder input, during the Definition phase of the project.

**Exhibit 7 – Quality Monitoring Metrics**

Performance Metric	Measurement Approach	Success Parameter
<b>Category: Time/Budget</b>		
All work is completed on time	Establish temporal milestones for key tasks and deliverables in project plan and compare with actual performance  Schedule Performance Index (SPI) is the ratio of value delivered to the planned value for the activities scheduled to be completed $SPI = EV / PV$	95% of deliverables meeting deadline; number of days beyond target less than 5% of scheduled development time  Schedule Performance Index less than 95% triggers flag and addressed monthly
All work completed within budget	Compare planned budget and performance against actual and calculate:  Cost Performance Index (CPI) = ratio of value delivered to the actual cost for the activities delivered $CPI = EV / AC$	Cost Performance Index less than 95% triggers flag and addressed monthly
<b>Category: Quality</b>		
Narratives flow in a logical manner	All deliverables subjected to documented project deliverable review process	95% compliance with review process, and noted in status report
Data quality is monitored to ensure clarity accuracy and consistency	All deliverables subjected to documented project deliverable review process	95 % compliance with review process and noted in status report
Deliverables are free of oversights in substance or omissions	All deliverables subjected to documented project deliverable review process	95% compliance with review process and noted in status report
The solution is adequate to project requirements	Use requirements traceability matrix (or automated tool) to map requirements to system design	Greater than 95% of requirements can be mapped to system design
<b>Category: Project Management</b>		
All work complies with program standards, guidelines, and policies	Produce monthly status reports that comply with program reporting standards, including burn rate, hours used, and hours remaining; note emerging risks and issues; provide mitigation plans  Quarterly financial reports are produced	Monthly reports produced by the 15 <sup>th</sup> day of the following month. Significant issues and flagged items reviewed in person  Produce financial reports within 15 days of the end of the quarter

Performance Metric	Measurement Approach	Success Parameter
All deliverables are prepared and presented consistent with normal business standards; written documents are clear and concise	All deliverables subjected to documented project deliverable review process	95% compliance with review process and noted in status report
No complaints without a resolution plan within 5 business days after matter brought to Program Manager's or Project Manager's attention	Employ a complaint tracking log; note complaints in status reports and date of resolution	Track complaint resolution time frames in monthly reports and address any outliers greater than 5 days  Provide explanation for delay and a plan to resolve the matter

### 3.6 Problem Tracking, Reporting, and Corrective Actions

The project team will utilize documented procedures to track, report, and apply corrective measures for project related problems. Problems and issues associated with the project will be documented via Bi-Weekly Project Status reports. All problems, project status, and review reports will be documented and maintained by the project team. The project team will maintain preventative and corrective action procedures.

To prevent issues, defects, and problems, the team will employ a formal risk and issue tracking processes:

- Risk Management** – The team will employ a risk management process to inform management and Team members about project risk areas and their potential consequences. Risks will be reported bi-weekly and tracked in the risk tracking template and evaluated in terms of their potential impact on meeting the target completion date (schedule), increasing project costs (cost), and decreasing quality of deliverables (quality).
- Issue Tracking** – The project team will document, track, and manage issues through an Issues Log. Each issue is assigned a priority, a person responsible for the issue, issue description, a resolution due date, and a status. During the bi-weekly status meetings, functional and technical issues and problems will be discussed to promptly address and resolve them before they impact the schedule and budget.
- Defect Tracking** – The project team will document software defects, as they are identified in the Testing activities. Each defect will be assigned a severity level, a description, identification date, and a status. Performance metrics will be tracked, including time to resolution, number of defects (identified by Testing activity and application component), and current status. Quality metrics will be established for each Testing activity, requiring successful achievement before the project progresses to the next activity.
- Quarterly Project Review** – The project team will meet with stakeholder technical and mission representatives every three months to review the project's progress, achievements, objectives and strategic vision.

## 1.7 Testing Methodology

To verify the quality of the project solution, the RIO project team will employ a disciplined, requirements-driven testing approach that integrates industry leading tools and standard processes and procedures. The project team develops a mutual stakeholder-project team understanding of requirements *up front*, and drives the testing process with those requirements to ensure end-user satisfaction with the product.

Key features of this testing methodology, contributing to its effectiveness, are as follows:

- **Requirements Traceability** – Verify that all requirements have been implemented. Map business requirements to test cases and test scenarios. This end-to-end traceability ensures the test team understands which parts of the solution implement the requirements and which tests are necessary to verify that the requirements have been implemented correctly.
- **Automated Defect Tracking** – Verify that issues are identified, tracked, and resolved in a timely manner.
- **Test Execution and Reporting Procedures** – Provide a framework for planning, executing, and reporting the results of all testing, both manual and automated. All testing work products (e.g., plans, scenarios, scripts, and data sets) products are placed under configuration and change control to ensure the integrity of the testing process. Testing processes will include unit testing, integration testing, system (end-to-end) testing, performance testing, and user acceptance testing.

The processes that support this testing methodology are described in CyberData Technologies' *Project Quality Management* method.

## Appendix – System Test Plan Template

1. Overview
  - 1.1 Introduction
  - 1.2 Project Objectives
  - 1.3 Quality Management Objectives
  - 1.4 Reference Documents
  
2. Testing Phase Definition
  - 2.1 Overview
  - 2.2 Testing Definition
  - 2.3 Testing Scope
  - 2.4 Out of Scope
  - 2.5 Project Assumptions
  
3. System Test Plan
  - 3.1 Overview
  - 3.2 Testing Metrics and Activities Gates
  - 3.3 Testing Schedule/WBS
  - 3.4 Functional Unit Testing
  - 3.5 Integration Testing
  - 3.6 System (End-to-End) Testing
  - 3.7 User Acceptance Testing
  - 3.8 Performance Testing
  - 3.9 Defect Tracking, Reporting, and Corrective Actions
  
4. Testing Management
  - 4.1 Project Team Organization
  - 4.2 Testing Roles and Responsibilities
  - 4.3 Test Environment Requirements
  - 4.4 Test Environment

4.5 Required Support and Communications

4.6 Tools and Configurations

4.7 Test Training Plan

4.8 Test Case Management

4.9 Test Data Requirements

4.10 Configuration Management

Appendix – Templates

Appendix – Project Testing Terms and Acronyms