# Table of Contents

Table of Contents.................................................................................................................. 2

About this Document.............................................................................................................. 4

Introduction To Quality Assurance......................................................................................... 5

- Everyday QA ...................................................................................................................... 5
- QA Concepts and Definitions ............................................................................................. 5
- Goals of Quality Assurance ............................................................................................... 6
- The Department of Quality Assurance ............................................................................... 6
- Summary ............................................................................................................................ 6

QA Activities.......................................................................................................................... 7

- Planning .............................................................................................................................. 7
- Tailoring Quality Assurance ............................................................................................... 7
- Creating the Quality Assurance Plan .................................................................................... 7
- Creating Test Cases ............................................................................................................ 8
- Establishing Standard and Procedures ............................................................................... 8
- Establishing Completion Criteria ....................................................................................... 8
- Auditing Against Standards and Procedures ...................................................................... 9
  - Monitoring Products and Processes ................................................................................. 9
  - Verification and Validation Monitoring ............................................................................ 9

QA Testing................................................................................................................................ 9

- Objectives of QA Testing .................................................................................................. 9
- Preparing for QA Testing .................................................................................................. 9
- The Key to Productive QA Testing .................................................................................... 10
- Twelve Types of QA Testing ............................................................................................. 10
- Workflow Diagram: Quality Assurance Testing Lifecycle .................................................. 13

Error Management.................................................................................................................. 14

- Objectives and Benefits .................................................................................................... 14
- Workflow Diagram: Defect Tracking .................................................................................. 14
- Error Review Team ............................................................................................................ 14
- ERT Plan of action .............................................................................................................. 15
- Verification and Classification of Errors .............................................................................. 15

QA and the Project Delivery Lifecycle ................................................................................... 17

- Common Project Problems ............................................................................................... 17
- Root Causes ......................................................................................................................... 17
- Missing Components ........................................................................................................... 17

QA Activities and Deliverables Within the Delivery Lifecycle ............................................. 18

- Assessment Phase ............................................................................................................. 18
- Planning Phase ................................................................................................................... 18
ABOUT THIS DOCUMENT

This document provides an overview of the concepts and functions of quality assurance, or QA, and provides guidelines for the implementation of a QA program.

This document also touches on each major activity within QA:

- Planning
- Establishing Standards and Procedures
- Establishing Completion Criteria
- Auditing Against Standards and Procedures
- Quality Assurance Testing
- Error Management

Appendices include:

- Configuration Management
- Controlled Testing Environment.

Throughout this document, "quality assurance” and “QA” are used interchangeably. Additionally, the terms “software” and "software development” refer to all software products, including websites and all website-related code.

This document is intended as a first draft of the Quality Assurance Guidelines for, and it is open for revision.
INTRODUCTION TO QUALITY ASSURANCE

EVERYDAY QA

Every software development, enhancement, or maintenance project includes some quality assurance activities. Even a simple, one-person development job has QA activities embedded in it, even if the programmer denies that "quality assurance" plays a part in what is to be done. Each programmer has some idea of how code should be written, and this idea functions as a coding standard for that programmer.

Similarly, each of us has some idea of how documentation should be written—this is a personal documentation standard. We proofread and revise our documents, and programmers review their products to make sure they meet their personal standards. These are QA reviews, or audits. Each programmer and writer tests or inspects his or her own work, and these are verification and validation processes.

A project's formal QA program includes the assurance processes that each team member goes through, but it involves planning and establishing project-wide standards, rather than relying on personal standards and processes. The extent and formality of project QA activities are decisions that the client, project manager, and the QA department make based on their assessment of the project and its risks.

QA CONCEPTS AND DEFINITIONS

Quality assurance is the planned and systematic set of activities that ensures that software processes and products conform to requirements, standards, and procedures.

- Processes include all of the activities involved in designing, developing, enhancing, and maintaining software.
- Products include the software, associated data, its documentation, and all supporting and reporting paperwork.

QA includes the process of assuring that standards and procedures are established and are followed throughout the software development lifecycle.

- Standards are the established criteria to which the software products are compared.
- Procedures are the established criteria to which the development and control processes are compared.

Compliance with established requirements, standards, and procedures is evaluated through process monitoring, product evaluation, audits, and testing.

The three mutually supportive activities involved in the software development lifecycle are management, engineering, and quality assurance.

- Software management is the set of activities involved in planning, controlling, and directing the software project.
Software engineering is the set of activities that analyzes requirements, develops designs, writes code, and structures databases.

Quality Assurance ensures that the management and engineering efforts result in a product that meets all of its requirements.

GOALS OF QUALITY ASSURANCE

Software development, like any complex development activity, is a process full of risks. The risks are both technical and programmatic; that is, risks that the software or website will not perform as intended or will be too difficult to operate/browse, modify, or maintain are technical risks, whereas risks that the project will overrun cost or schedule are programmatic risks.

The goal of QA is to reduce these risks. For example, coding standards are established to ensure the delivery of quality code. If no standards are set, there exists a risk that the code will not meet the usability requirements, and that the code will need to be reworked. If standards are set but there is no explicit process for assuring that all code meets the standards, then there is a risk that the code base will not meet the standards. Similarly, the lack of an Error Management and Defect Life Cycle workflow increases the risk that problems in the software will be forgotten and not corrected, or that important problems will not get priority attention.

The QA process is mandatory in a software development cycle to reduce these risks, and to assure quality in both the workflow and the final product. To have no QA activity is to increase the risk that unacceptable code will be released.

THE DEPARTMENT OF QUALITY ASSURANCE

QA is an activity that should be organizationally independent of the producing organizations. QA functions are best performed in an discrete QA testing environment by organizational entities that are separate from the ones doing engineering or management activities. Administratively, the QA organization should report to top corporate management and interface with the project manager.

The reason for this separation of function is that the QA organization is the arm of management that assures that standards are met and that procedures are followed. If QA is not independent of the development activity, clear and impartial assessment will be difficult. Additionally, organizational independence helps ensure that testing will be requirements-driven and not influenced by the design or coding details.

Staff devoted purely to QA activities is usually small compared to the project staff, but it is important to have people with specific QA responsibilities. Too often, the axiom “quality is everybody's business” becomes “quality is nobody's business” if specific QA responsibilities are not assigned.

SUMMARY

QA is an essential part of the development and maintenance of software. It is part of the triad of activities, along with software management and software engineering, that determines the success of any software development, enhancement, or maintenance activity.
Planning

TAILORING QUALITY ASSURANCE

Specific project characteristics and risks influence Quality Assurance needs, and every Quality Assurance plan should be tailored to its project. Characteristics that should be considered include mission criticality of the project, schedule and budget, size and complexity of the product to be produced, and size and organizational complexity of the development staff.

The relationship of criticality to QA functions is, as one would expect: the more critical the project, the more significant and formal the Quality Assurance effort must be. However, the relationship of schedule and budget is not as intuitive: the tighter the budget and schedule, the more critical it is to have a well-planned and effective Quality Assurance effort. This does not mean that projects with more resources can afford to be lax, it just means that tight resources increase risks that should be offset by a strong Quality Assurance program.

The projected size of the end product influences the level of Quality Assurance required. A large project requires explicit and detailed standards for all of the products in order to get at least a minimum standard of quality from the varied ideas and experience of many different programmers. In addition, a large project requires significant efforts in testing and other verification activities, which must be established in the QA plan. For such a project, a significant and formal Quality Assurance program must be established or risks of poor quality products must be accepted.

On the other hand, a small project may require little formal Quality Assurance, and on a very small one, the Quality Assurance efforts may be left to the programmer involved if adequate informal planning is completed with guidance from the QA department.

CREATING THE QUALITY ASSURANCE PLAN

The purpose of creating a Quality Assurance plan is to document and specify the activities that will comprise Quality Assurance for a specific project. Using information about the project and a knowledge of the available Quality Assurance resources, the Quality Assurance project lead develops a plan tailored to the project.

An effective Quality Assurance program requires planning and follow-through. A Quality Assurance plan cannot simply evolve along with the project. Adequate Quality Assurance planning ensures that all Quality Assurance activities are focused on the unique requirements and risks associated with the project.

The QA project lead should consider the following guidelines when creating a QA Plan:

- Plan QA in conjunction with management and engineering planning, i.e., during the project concept and initiation phase.
- Phase QA activities properly. For example, design standards must be produced well before design is to be done.
- Complete tool development or procurement before the tools are needed. Especially important are the test tools and test data sources.
Creating Test Cases

A test case is a document that describes an input, action, or event and an expected response. A QA engineer or other tester performs the test case scenario to determine if a specific feature of an application is working correctly.

The process of developing test cases can help find problems in the requirements or design of an application since test case development requires thinking through the complete operation of the application. For this reason, it is useful to prepare test cases early in the development cycle.

A test case contains:
- Detailed test set-up instructions.
- Detailed test procedure, including all steps or actions.
- Expected results and pass/fail criteria.

Establishing Standard and Procedures

Establishing standards and procedures for software development is critical, since these provide the framework from which the software evolves. Standards, the established criteria to which the software products are compared, and procedures, the established criteria to which the development and control processes are compared, provide the prescribed methods for developing software. The role of QA is to ensure existence and adequacy of standards and procedures.

Proper documentation of standards and procedures is necessary since the QA activities of process monitoring, product evaluation, and auditing rely upon unequivocal definitions by which to measure project compliance.

- Documentation standards specify form and content for planning and product documentation and provide consistency throughout a project.
- Design standards specify the form and content of the design product. They provide rules and methods for translating the software requirements into the software design and for representing it in the design documentation.
- Documented procedures are explicit steps to be followed in carrying out a process. Examples of processes for which procedures are needed are configuration management, nonconformance reporting and corrective action, testing, and formal inspections.

Establishing Completion Criteria

Because of the nature of software, it is difficult to ascertain the status of a development or maintenance activity. It is important, therefore, to define criteria for the completion of specific development stages. During the implementation phase, one has to complete the lowest level (most detailed) design of small program elements, code the elements, and unit test them. When a significant number of program elements are involved, it is difficult for anyone to ascertain the status of the units without specific completion criteria. For example, if there is a criterion that detailed design is complete only after the rework that follows a design inspection, then the design can be said to be either complete or incomplete depending on the status of the rework.
The establishment of completion criteria is a management activity, but the audit of status records is an important QA activity. Audits determine the accuracy of the reported status.

**Auditing Against Standards and Procedures**

A fundamental QA technique is the audit, which looks at a process or a product in depth and compares it to established procedures and standards. Audits are used to review management, technical, and QA processes to provide an indication of the quality and status of the software product.

The purpose of a QA audit is to assure that proper control procedures are being followed, that required documentation is maintained, and that the developer's status reports accurately reflect the status of the activity. The QA deliverable is an audit report to management consisting of findings and recommendations to bring the development into conformance with standards and/or procedures.

**MONITORING PRODUCTS AND PROCESSES**

Product evaluation and process monitoring assure that the software development and control processes described in the project's management plan are correctly carried out and that the project's procedures and standards are followed. Products are monitored for conformance to standards, and processes are monitored for conformance to procedures. Audits are a key technique used to perform product evaluation and process monitoring. Review of the management plan should ensure that appropriate QA approval points are built into these processes.

Ideally, the first products monitored by QA should be the project's standards and procedures. QA assures that clear and achievable standards exist and then evaluates compliance of the software product to the established standards. Functional and technical document evaluations follow. This assures that the software product reflects the requirements and standards as identified in the management plan.

**VERIFICATION AND VALIDATION MONITORING**

QA assures Verification and Validation (V&V) activities by monitoring and participating in product reviews, inspections, and walk-throughs. A review looks at the overall picture of the product being developed to see if it satisfies its requirements. In formal reviews, actual work done is compared with established standards. Reviews are part of the development process, and they should be conducted at the end of each phase of the lifecycle to identify problems and determine whether the interim product meets all applicable requirements. The results of the review should provide a ready/not-ready decision to begin the next phase. Although the decision to proceed is a management decision, QA is responsible for advising management and participating in the decision.

**QA Testing**

Software testing verifies that the software meets its requirements and that it is complete and ready for delivery.

**OBJECTIVES OF QA TESTING**

- Assure the quality of client deliverables.
- Design, assemble, and execute a full testing lifecycle.
- Confirm the full functional capabilities of the final product.
- Confirm stability and performance (response time, etc.) of the final product.
- Confirm that deliverables meet client expectations/requirements.
- Report, document and verify code and design defects.

**PREPARING FOR QA TESTING**

Prior to conducting formal software testing, QA develops testing documentation (including test plans, test specifications, and test procedures) and reviews the documentation for completeness and adherence to standards. QA confirms that:

- The test cases are testing the software requirements in accordance with test plans.
- The test cases are verifiable.
- The correct or "advertised" version of the software is being tested (by QA monitoring of the Configuration Management activity).

QA then conducts the testing in accordance with procedure, documents and reports defects, and reviews the test reports.

**THE KEY TO PRODUCTIVE QA TESTING**

It is crucial to recognize that all testing will be conducted by comparing the final product to the product’s set requirements; therefore, product requirements must state all functionality of the software and *must be updated as changes are made*. Any functionality that does not meet the requirements will be recorded as a defect until resolution is delivered.

**TWELVE TYPES OF QA TESTING**

1. **Unit testing (conducted by Development)**

   Unit test case design begins after a technical review approves the high level design. The unit test cases shall be designed to test the validity of the program's correctness. White box testing is used to test the modules and procedures that support the modules. The white box testing technique ignores the function of the program under test and focuses only on its code and the structure of that code. To accomplish this, a statement and condition technique shall be used. Test case designers shall generate cases that not only cause each condition to take on all possible values at least once, but that cause each such condition to be executed at least once. In other words:

   - Each decision statement in the program shall take on a true value and a false value at least once during testing.
   - Each condition shall take on each possible outcome at least once during testing.

2. **Configuration Management**
The configuration management team prepares the testing environment

3. **Build Verification**

When a build has met completion criteria and is ready to be tested, the QA team runs an initial battery of basic tests to verify the build.

- If the build is not testable at all, then the QA team will reject the build.
- If portions of the website are testable and some portions are not yet available, the project manager, technical lead and QA team will reassign the build schedule and deliverable dates.
- If all portions of the build pass for testing, the QA team will proceed with testing.

4. **Integration Testing**

Integration testing proves that all areas of the system interface with each other correctly and that there are no gaps in the data flow. The final integration test proves that the system works as an integrated unit when all the fixes are complete.

5. **Functional Testing**

Functional testing assures that each element of the application meets the functional requirements of the business as outlined in the requirements document/functional brief, system design specification, and other functional documents produced during the course of the project (such as records of change requests, feedback, and resolution of issues).


Non-functional testing proves that the documented performance standards or requirements are met. Examples of testable standards include response time and compatibility with specified browsers and operating systems.

If the system hardware specifications state that the system can handle a specific amount of traffic or data volume, then the system will be tested for those levels as well.

7. **Defect Fix Validation**

If any known defects or issues existed during development, QA tests specifically in those areas to validate the fixes.

8. **Ad Hoc Testing**

This type of testing is conducted to simulate actual user scenarios. QA engineers simulate a user conducting a set of intended actions and behaving as a user would in case of slow response, such as clicking ahead before the page is done loading, etc.

9. **Regression Testing**
Regression testing is performed after the release of each phase to ensure that there is no impact on previously released software. Regression testing cannot be conducted on the initial build because the test cases are taken from defects found in previous builds.

Regression testing ensures that there is an continual increase in the functionality and stability of the software.

10. **Error Management**

During the QA testing workflow, all defects will be reported using the error management workflow.

Regular meetings will take place between QA, system development, interface development and project management to discuss defects, priority of defects, and fixes.

11. **QA Reporting**

QA states the results of testing, reports outstanding defects/known issues, and makes a recommendation for release into production.

12. **Release into production**

If the project team decides that the build is acceptable for production, the configuration management team will migrate the build into production.
**Workflow Diagram: Quality Assurance Testing Lifecycle**

1. **Build 1**
   - **Build Rejected**: All Failed
   - **Build Verification Test**
     - **Passed All**: Reassign build schedules and deliverables.
     - **Some Passed**: Defect Fix Verification (if available)
     - **Performance Testing** (Standards are met, non-functional)
     - **Integration Testing** (Business Scenarios)
   - **Config. Mgmt. Prepares Environment**
   - **Passed All**: Functional Testing (Functional Req. are met)
   - **Defect Fix Verification** (if available)

2. **Build x**
   - **Build Rejected**: All Failed
   - **Build Verification Test**
     - **Passed All**: Reassign build schedules and deliverables.
     - **Some Passed**: Defect Fix Verification (if available)
     - **Performance Testing** (Standards are met, non-functional)
     - **Integration Testing** (Business Scenarios)
   - **Config. Mgmt. Prepares Environment**
   - **Passed All**: Functional Testing (Functional Req. are met)
   - **Defect Fix Verification** (if available)

**Testing Complete**
- **Config. Mgmt. Deploys to Production Staging.**
- **A QA report will be delivered stating the results of testing and Outstanding defects/known issues**
- **Passed All**: Integration Testing (Business Scenarios)
- **Passed All**: Regression Testing (Standards are met, non-functional)

All defects will be reported and processed using the Error Management Workflow. There will be regular meetings between QA, System Dev, Interface Dev, and Project manager to discuss defects and priority for the fix. Additionally, a regular progress report will be provided from all parties on their deliverables.
Error Management

An sound error management process ensures maximum efficiency for defect recognition and resolution during software development and deployment to production.

OBJECTIVES AND BENEFITS

- Maintain a defect tracking system to reliably monitor defects and fixes.
- Preserve a history of defects and their fixes.
- Ensure prompt and efficient identification and notification of defects
- Provide timely fixes and deployment

WORKFLOW DIAGRAM: DEFECT TRACKING
**ERROR REVIEW TEAM**

The above workflow addresses only defects that are identified during development; it does not address the identification and resolution of production defects that are reported to or discovered. Production defects are errors that are identified in “live,” released software.

Because these defects hold high priority and need the most attention, an error review team (ERT) must be assigned to all live products. An ERT provides a prompt and efficient response to defect recognition in a production environment and ensures a timely resolution. The team will consist of the following members:

- Development team leader
- Project manager
- QA team lead
- Client representative

**ERT PLAN OF ACTION**

When a defect in a production application is reported, the project ERT executes the following plan of action within four hours from the time the bug is reported:

1. All members of the ERT team are notified as soon as the defect is reported.

2. An immediate meeting is called with ERT and the client to discuss the details of the error, the environment, and the steps necessary to reproduce the defect. Known errors are reviewed to ensure that the defect is not a duplicate.

3. The defect is reproduced in-house or in a staging environment.

4. If valid, the defect is entered in the defect tracking system as OPEN.

5. The client is notified of the validity of the error.

6. Prioritization of valid errors is conducted.

7. Appropriate developers are notified.

8. A short-term resolution is discussed. (“How can we get the system functional ASAP?”)

9. A long-term resolution is discussed. (“What permanent changes need to implemented to prevent errors like this?”)

10. Client is contacted with the short- and long-term resolutions and the timelines.
11. The short-term resolution is implemented (Defect is marked FIXED in defect tracker with the resolution). Upon verification of the defect fix, the defect is marked as VERIFIED and deployed to production. Upon successful deployment and confirmation from the client, the defect is marked CLOSED.

12. Long-term resolution is implemented, verified, and deployed.

**Verification and Classification of Errors**

The ERT plan of action calls for the team to classify and prioritize the error once it confirms the validity of a production defect. Valid errors will be classified as follows:

- **Critical**: a serious error that is either a showstopper or of such importance as to radically affect the functionality of the system. Examples include:
  - A user is unable to complete a transaction because of a consistent crash during processing
  - Incorrect data is being passed resulting in heavy corruption or system crashes

- **Major**: an error where a less important element of functionality is affected, or data which does not have a major impact is corrupted or lost. Examples include:
  - A value that is not defaulting correctly must be input manually.
  - A process is not being completed as intended, but there is an alternative method of completing the process

- **Minor**: a minor error that does not hinder functionality. These types of errors are mainly cosmetic bugs.

Once the defect is classified and the development team has been notified, developers should provide a fix for "critical" errors within 24 hours. This is the turn-around time from when the defect is discussed at the ERT review meeting to when fix is released to the system test environment. (In the event that a critical error prevents the system from functioning, the turn-around should be significantly less.) "Major" errors should be turned around in 36 hours; and "minor" errors should be turned around in 48 hours.
Common Project Problems

Like all software development activities, projects risk the following common technical and programmatic problems:

- Inaccurate understanding of the project requirements.
- Inflexibility; inability to adapt to changing requirements.
- Modules that do not work together.
- Late discovery of serious project flaws.
- Scant record of who changed what, when, or why.
- Limited roll-back capabilities

ROOT CAUSES

Such problems often stem from the following root causes:

- Lack of communication; poor information work flow.
- Lack of processes.
- Lack of standards and/or procedures.
- Lack of process for integration (the big picture).
- Lack of solid testing methodology.

MISSING COMPONENTS

With the following programmatic components in place, the root causes of many common project problems may be corrected:

- Quality assurance
- Configuration management
- Version control
- Controlled testing environment
- Quality assurance testing
- Error management
Standardized work flows for all of the components above

**QA Activities and Deliverables Within the Delivery Lifecycle**

Each of the five phases of Project Delivery Lifecycle will incorporate QA activities and deliverables that off-set the risks of common project problems. This summary of the Project Delivery Lifecycle incorporates a high-level list of the QA activities and deliverables associated with each phase.

**Assessment Phase**

Assessment process consists of market research and a series of structured workshops that the and client teams participate in to discuss and analyze the project objectives and develop a strategic plan for the effort. The products of these meetings, combined with market research, form the basis for the final output of the assessment: a tactical plan for realizing specific business and project objectives.

- **QA Deliverables**
  - QA Editor submits revised and approved deliverable documents.

**Planning Phase**

In the Planning phase, the team defines specific system requirements and develops strategies around the information architecture (static content and information flows) and the business functions that will be addressed.

- **QA Activities**
  - **Establishing Standards and Procedures:** QA records the set requirements.
  - **Planning (Test Matrix):** QA develops a test matrix. QA confirms that all set requirements are testable and coincide with the project objectives.
  - **Auditing Against Standards and Procedures:** QA editor edits the documents and confirms that they meet the objectives and the quality standards for documents.
  - **Establishing Completion Criteria:** QA records the completion criteria for the current phase.

- **QA Deliverables**
  - QA submits an initial test matrix.
  - QA Editor submits revised and approved deliverable documents.
DESIGN PHASE

During the Design phase, the team identifies all of the necessary system components based on the requirements identified during the Assessment and Planning phases. The team then creates detailed design specifications for each component and for the associated physical data requirements.

- **QA Activities**
  - **Auditing Standards and Procedures:** QA confirms that all designs meet the set requirements and notes any discrepancies. Additionally, QA identifies any conflicts or discrepancies between the final design of the system and the initial proposal for the system and confirms that an acceptable resolution has been reached between the project team and the client.
  - **Planning (QA Plan, QA Test Plan):**
    - QA begins developing the QA Plan.
    - QA revised the test matrix to reflect any changes and/or additions to the system.

- **QA Deliverables**
  - QA presents the initial QA test plan.
  - QA submits a revision of the test matrix.

DEVELOPMENT PHASE

During the Development phase, the team constructs the components specified during the Design Phase.

- **QA Activities**
  - **Planning (Test Cases):** Using the test matrix, QA develops a set of test cases for all deliverable functionality for the current phase.
  - **Prepare for Quality Assurance Testing:**
    - QA confirms that all test cases have been written according to the guidelines set in the QA test plan.
    - Quality Assurance works closely with the Configuration Management group to prepare a test environment.

- **QA Deliverables**
  - QA submits a set of Test Cases.
  - QA Environment is set up.
IMPLEMENTATION PHASE

In the Implementation phase, the team focuses on testing and review of all aspects of the system. The team will also develop system documentation and a training or market test plan in preparation for system launch.

❖ QA Activities
  ♦ QA Testing: QA executes all test cases in the QA testing cycle.

❖ QA Deliverables
  ♦ Test Results
  ♦ Defect Reports
Workflow Diagrams: the Project Delivery Lifecycle

Workflow Without QA Activities

Proposal → IRNA

Tech. Research → IRNA → Final Project Definition → Final Project Def. Approved

Tech. Research → System Design → Phase X → Test Planning → Test Execution

Project Definition → Final Project Definition → Final Project Def. Approved

Interface Research → Interface Design

System Design → Interface Design

Phase X
WORKFLOW WITH QA ACTIVITIES

Proposal → IRNA

Tech. Research → Project Definition → Quality Assurance Editor

Research → Final Project Definition

System Design → Test Plan → Quality Assurance Proj. Lead

System Develop. → Test Cases written to conform Requirements. All Requirements Must be updated if changes are made to the system.

Interface Design → Quality Assurance Proj. Lead

Test Cases written to conform Requirements. All Requirements Must be updated if changes are made to the interface.

Development

Follow Configuration management Workflow to move files from QA to Development

Defects are reported.

Follow Error management Workflow for the life cycle of all defects

Follow Quality Assurance Testing Workflow to conduct all testing.

Build 1

QA will conduct a Smoke Test before executing the Test Cases. The purpose is to determine if the delivered build holds all build functionality and can be tested in detail.

Build 2/ Defect fix from build 1

QA will conduct a Smoke Test before executing the Test Cases. The purpose is to determine if the delivered build holds all build functionality and can be tested in detail.

Test Case Execution For Build1

Test Case Execution For Build2
Appendix A: Configuration Management

The configuration management (CM) team is responsible for maintaining the validity of code as it moved from one environment to the next. The CM team takes part in the development workflow every time there is a change of code status. For example, when a software build phase is completed and ready to be tested, the code must be moved from the development environment to testing environment. This task belongs to the CM team.

CM follows a controlled workflow to promote and deploy all code. Additionally, CM is responsible for maintaining all source control databases (SCD) and the standards and processes related to SCD.

OBJECTIVES AND BENEFITS

- Improve the complete software development and maintenance cycle.
- Make QA testing easier, reliable, and effective.
- Remove error-prone steps from product release management.
- Support and ease the challenges of change management and defect tracking.
- Improve the quality and validity of code through version control.
- Provide code roll-back if necessary, or trace related code components.
- Maintain a history of code changes (who, what, when and why).

CM RESPONSIBILITIES

The CM team provides the following services throughout the course of product development and deployment:

- Implement and maintain version control in development and testing environments.
- Prepare and configure QA servers per the specifications provided by system designers.
- Move code into the testing environment (servers and SCD) and establish the initial (or “baseline”) code base.
- Give the go-ahead to QA to commence testing after the code is moved.

SIMPLIFIED CM WORKFLOW

CM’s role in software development and deployment may be summarized as follows:

- In preparation for QA testing, CM moves all labeled files from the development SCD to the QA servers and the QA SCD, where they are labeled again.
During QA testing, CM moves all new builds and defect fixes to the QA SCD and QA servers and labels them.

When the files are approved, CM labels and moves them from the QA SCD into the production staging servers.

Upon approval from project management and QA, CM promotes the final code base to production servers for deployment.

Any defect fixes or enhancements after the initial build will also be deployed following the above steps.

**QA AND CONFIGURATION MANAGEMENT**

QA audits all CM functions for adherence to CM plans and procedures and prepares reports of its findings. QA reviews the CM plans for compliance with software CM policies and verifies that the plans include follow-up for defects.

The CM functions monitored and audited by QA include baseline control, configuration identification, and configuration control. QA assures that:

- Baselines are established and consistently maintained for use in subsequent baseline development and control.

- Approved changes to baseline software are made properly and consistently in all products, and no unauthorized changes are made.
Diagram: Configuration Management

Code from QA SS is moved to Production SS and Staging Servers.

Development Process:
- Config. Management builds all files.
- QA cycle.
- Quality Assurance.
- Code from QA SS moved to Production SS.

Code from QA SS is moved to Production SS and Staging Servers.

QA and Release Engineering need to know what is under development and when (approximately) will it be finished.

Cycle to repeat as needed before a release.

Reports are generated for new/modified/deleted files and the reason for the change (enhancement/new component/defect fix).

Release Engineer needs to be aware of all changes in the Dev Servers and all possible changes to be moved to QA for testing.

Defects are reported to QA SS and QA Servers.

Dev Checkin/Checkout.
- Development - Source Safe.
- QA - Source Safe.
- Production - Source Safe.

Deployment Team.
- Production Site.
- Staging Site.
- Code for the Phase is moved to the servers.
- Deployment.


Source Safe.

Source Safe.

Source Safe.

Source Safe.

Source Safe.

Source Safe.

Source Safe.

Source Safe.
Appendix B: Controlled Testing Environment

OVERVIEW

A controlled testing environment is a simulated production environment to observe the code and its behavior. This environment, in which QA will conduct the testing lifecycle, will be created and maintained independently of the development environment.

This environment must be isolated and independent from the corporate network in order to conduct controlled testing. All stress testing, for instance, needs to be limited to the testing servers to avoid stressing the corporate network and to ensure a true test of the desired servers.

There are numerous benefits of having a separate testing environment. For instance, development of future components may continue while existing builds are tested independently.

REQUIREMENTS

A controlled testing environment must include one box with access to the servers as well as to the corporate network. This box will contain the source code (Microsoft Source Safe, or SS). The configuration management team needs full access to this SS box in order to move code from the development environment to the testing environment. All code in the servers will be maintained in the SS box and labeled accordingly.

The servers must also have dial-up access in order to conduct download testing, as the majority of users are not using DSL or other fast Internet connections. Though there are many tools that estimate this time, such estimates cannot replace tests of live behavior.

Numerous test machines are an important part of the controlled test environment. The QA team will conduct stress tests and OS and browser compatibility tests using these test machines.

ESTIMATED NEED FOR EQUIPMENT

<table>
<thead>
<tr>
<th>Hardware</th>
<th>Quantity</th>
<th>Basic Software</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hub</td>
<td>1</td>
<td>N/A</td>
</tr>
<tr>
<td>Server</td>
<td>2</td>
<td>NT, IIS, ...</td>
</tr>
<tr>
<td>Server</td>
<td>1</td>
<td>Unix, Apache</td>
</tr>
<tr>
<td>Server</td>
<td>1</td>
<td>Solaris, ATG Dynamo</td>
</tr>
<tr>
<td>Server</td>
<td>1</td>
<td>Solaris, Oracle</td>
</tr>
<tr>
<td>Server</td>
<td>1</td>
<td>NT, SQL</td>
</tr>
<tr>
<td>Stress PC (max RAM)</td>
<td>1</td>
<td>Windows, stress test software, browsers</td>
</tr>
<tr>
<td>Test machine</td>
<td>2</td>
<td>Windows, testing software, browsers</td>
</tr>
<tr>
<td>Test machine</td>
<td>1</td>
<td>Mac OS, testing software, browsers</td>
</tr>
<tr>
<td>Removable hard drives</td>
<td>10</td>
<td>Different OS/browsers/software</td>
</tr>
<tr>
<td>Source control PC</td>
<td>1</td>
<td>Windows, Microsoft Source Safe</td>
</tr>
<tr>
<td>-------------------</td>
<td>---</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Monitor/keyboard/mouse</td>
<td>6</td>
<td>N/A</td>
</tr>
<tr>
<td>Omni View</td>
<td>1</td>
<td>N/A</td>
</tr>
</tbody>
</table>

**Diagram: Controlled Testing Environment Specifications**

- Corporate Network
- QA Team Personal Machines
- Gate into Environment
- Source Control Database
- QA Stress test PC
- QA Test PC
- QA Test Mac
- NT Box w/ IIS
- NT Box w/ IIS
- NT SQL Box
- Solaris w/ Apache
- Solaris w/ Red Hat
- Sun Box w/ Oracle
- Solaris w/ ATG
- Removable Hard Drive
- Removable Hard Drive
- Removable Hard Drive
- Dial-Up Access