

**Software Testing and Quality
Assurance White Papers**

Pointe Technology Group, Inc.

Software Testing and Quality Assurance White Papers

Table of Contents

TABLE OF CONTENTS	1
SOFTWARE TESTING AND SOFTWARE QUALITY ASSURANCE	1
QA Services	1
Process and Procedures – Why follow them?	1
Standards and Templates – What is supposed to be in a document?.....	1
Test Readiness Review – Who is ready for the next phase?	2
Different Levels of Testing.....	2
Unit Testing.....	2
Parallel/Audit Testing.....	2
Functional Testing.....	2
Usability Testing	3
Incremental Integration Testing	3
Integration Testing	3
System Testing	3
End-to-End Testing	3
Regression Testing	3
Sanity Testing.....	4
Performance Testing.....	4
Load Testing.....	4
Installation Testing	4
Security/Penetration Testing	4
Recovery/Error Testing	4
Compatibility Testing.....	4
Comparison Testing	4
Acceptance Testing	4
Alpha Testing	5
Beta Testing.....	5
Resources.....	5
Test/QA Team Lead	5
Testers	5
Test Build Manager/System Administrator/Database Administrator.....	5
Technical Analyst & Test Configuration Manager	5
SOFTWARE TESTING METHODOLOGY	6
Step 1 - Create Test Strategy	7
Step 2 - Create Test Plan/Design	7

Pointe Technology Group, Inc.

Software Testing and Quality Assurance White Papers

Step 3 - Execute Tests.....	9
FREQUENTLY ASKED QUESTIONS	11
Why is it often hard for management to get serious about quality assurance?	11
Why does software have bugs?.....	11
How can new Software QA processes be introduced in an existing organization?	12
What is verification? validation?	12
What is a 'walkthrough'?.....	12
What's an 'inspection'?	12
What are five common problems in the software development process?.....	13
What are five common solutions to software development problems?	13
What is software 'quality'?	13
What is 'good code'?.....	14
What is 'good design'?	14
What is the 'software life cycle'?	14
Will automated testing tools make testing easier?.....	14
What makes a good test engineer?.....	15
What makes a good Software QA engineer?.....	15
What makes a good QA or Test manager?	15
What's the role of documentation in QA?.....	15
What's the big deal about 'requirements'?	16
What's a 'test plan'?	16
What's a 'test case'?	16
What should be done after a bug is found?.....	17
What is configuration management (CM)?.....	17

Pointe Technology Group, Inc.

Software Testing and Quality Assurance White Papers

What if the software is so buggy it can't really be tested at all?	17
How can it be known when to stop testing?	17
What if there isn't enough time for thorough testing?	18
What if the project isn't big enough to justify extensive testing?.....	18
What can be done if requirements are changing continuously?	19
What if the application has functionality that wasn't in the requirements?	19
How can Software QA processes be implemented without stifling productivity?	20
What if organization is growing fast that fixed QA processes are impossible?	20
How does a client/server environment affect testing?	20
How can Web sites be tested?.....	21
How is testing affected by object-oriented designs?	22
Why is it recommended to test during the design phase?	22
What are the five approaches to integration testing?	23
What are the three software development process models?	23
GLOSSARY OF TERMS	24
CONTACT.....	39
BIBLIOGRAPHY.....	39

Software Testing and Software Quality Assurance

Software Quality Assurance (QA) at Pointe Technology is oriented to 'prevention'. It involves the entire software development process. Process is monitoring and improving the process, making sure that any agreed-upon standards and procedures are followed, and ensuring that problems are found and dealt with.

Software Testing at Pointe Technology is oriented to 'detection'. Testing involves the operation of a system or application under controlled conditions and evaluating the results.

Organizations vary considerably in how they assign responsibility for QA and testing. Sometimes they are the combined responsibility of one group or individual. Also common are project teams that include a mix of testers and developers who work closely together, with overall QA processes monitored by project managers. It will depend on what best fits an organization's size and business structure.

Pointe can provide QA and/or Software Testing services. This document details some aspects of how we can provide quality and web site testing services. For more information please contact one of the Pointe Players listed in the last section of this document.

QA Services

QA ensures that all parties concerned with the project adhere to the process and procedures, standards and templates, and test readiness reviews.

Our QA service depends on the customers and projects. A lot will depend on team leads or managers, feedback to developers, and ensuring adequate communications among customers, managers, developers, and testers.

Process and Procedures – Why follow them?

Detailed and well-written processes and procedures ensure that the correct steps are being executed to facilitate a successful completion of a task. They also ensure that a process is repeatable.

Once Pointe Technology has learned and reviewed your processes and procedures we will recommend improvements and/or additions. Once you approved and signed off, CM will enter them into the version control tool. Then a plan will be devised to 1) document, 2) implement, and 3) monitor these processes and procedures.

Processes and procedures will be reviewed and updated periodically by QA and the process owner to ensure that they are current and relevant.

Standards and Templates – What is supposed to be in a document?

All documents should be written to a certain standard and template. This gives documents uniformity. It also helps in learning where information is located making it easier for a user to find what they want. Lastly, with standards and templates information will not be accidentally omitted from a document.

Pointe Technology Group, Inc

Software Testing and Quality Assurance White Papers

Once the Pointe Technology has learned and reviewed your standards and templates we will recommend improvements and/or additions. Once they are approved and signed off by you, CM will enter them into the version control tool. The standards and templates will be enforced by QA during: all levels of testing.

Standards and templates will be reviewed and updated periodically by the process owners to ensure that they are current and relevant.

Test Readiness Review – Who is ready for the next phase?

The test readiness review ensures that all artifacts are complete and correct & information is being passed on to the next group. The group “passing off” must follow an Exit Criteria. The group “receiving “ must follow an Entrance Criteria. The QA person ensures that all parties are prepared and that the process is correctly performed.

Pointe can suggest an Entrance and Exit Criteria for review, approval, and eventually sign off. Once Pointe and the customer have agreed upon the Entrance and Exit Criteria CM will enter it into the version control tool. QA will then enforce these criteria during every Test Readiness Review.

Both Entrance and Exit Criteria will be reviewed and updated periodically by QA and the process owner to ensure that they are current and relevant.

Different Levels of Testing

Pointe has expertise in testing at all of the below testing levels. At each test level we will document the results. It is recommended to get sign-off and check in all documentation and code according to Configuration Management (CM) procedures to ensure quality testing.

Each level of testing is either considered black or white box testing.

- Black box testing: not based on any knowledge of internal design or code. Tests are based on requirements and functionality.
- White box testing: based on knowledge of the internal logic of an application's code. Tests are based on coverage of code statements, branches, paths, and conditions.

Unit Testing

Unit Testing is the first level of dynamic testing and is first the responsibility of the developers than the testers. After the expected test results are met or differences are explainable/acceptable.

Parallel/Audit Testing

Testing where the user reconciles the output of the new system to the output of the current system to verify the new system does the operations correctly.

Functional Testing

Black-box type of testing geared to functional requirements of an application. Testers should do this type of testing.

Usability Testing

Testing for 'user-friendliness'. Clearly this is subjective, and will depend on the targeted end-user or customer. User interviews, surveys, video recording of user sessions, and other techniques can be used. Programmers and testers are usually not appropriate as usability testers.

Incremental Integration Testing

It is recommended that continuous testing of an application as new functionality is added. This may require that various aspects of an application's functionality be independent enough to work separately before all parts of the program are completed, or that test drivers be developed as needed; done by programmers or by testers.

Integration Testing

Upon completion of unit testing, integration testing, which is black box testing, will begin. The purpose is to ensure that distinct components of the application still work in accordance to customer requirements. Test sets will be developed with the express purpose of exercising the interfaces between the components. This activity is to be carried out by the Test Team. Integration test will be termed complete when actual results and expected results are either in line or differences are explainable/acceptable based on client input.

System Testing

Upon completion of integration testing, the Test Team will begin system testing. During system testing, which is a black box test, the complete system is configured in a controlled environment to validate its accuracy and completeness in performing the functions as designed. The system test will simulate production in that it will occur in the "production-like" test environment and test all of the functions of the system that will be required in production. The Test Team will complete the system test.

Prior to the system test, the unit and integration test results will be reviewed by SQA to ensure that all problems have been resolved. It is important for higher level testing efforts to understand unresolved problems from the lower testing levels. System test is deemed complete when actual results and expected results are either in line or differences are explainable/acceptable based on client input.

End-to-End Testing

Similar to system testing; the 'macro' end of the test scale; involves testing of 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.

Regression Testing

The objective of regression testing is to ensure that software remains intact. A baseline set of data and scripts will be maintained and executed to verify that changes introduced during the release have not "undone" any previous code. Expected results from the baseline are compared to results of the software being regression tested. All discrepancies will be highlighted and accounted for before testing proceeds to the next level.

Pointe Technology Group, Inc

Software Testing and Quality Assurance White Papers

Sanity Testing

Sanity testing will be performed whenever cursory testing is sufficient to prove that the application is functioning according to specifications. This level of testing is a subset of regression testing. It will normally include a set of core tests of basic GUI functionality to demonstrate connectivity to the database, application servers, printers, etc.

Performance Testing

Although performance testing is described as a part of system test, it can be regarded as a distinct level of testing. Performance testing will verify the load, volume, and response times as defined by requirements.

Load Testing

Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.

Installation Testing

Test full, partial, or upgrade install/uninstall processes. The installation test for a release will be conducted with the objective of demonstrating production readiness. This test is conducted after the application has been migrated to the client's site. It will encompass the inventory of configuration items (performed by the application's System Administration) and evaluation of data readiness as well as dynamic tests focused on basic system functionality. When necessary, a sanity test will be performed following the install.

Security/Penetration Testing

Test how well the system protects against unauthorized internal or external access, willful damage, etc; may require sophisticated testing techniques.

Recovery/Error Testing

Test how well a system recovers from crashes, hardware failures, or other catastrophic problems.

Compatibility Testing

Test how well software performs in a particular hardware/software/operating system/network/etc. environment.

Comparison Testing

Compare software weaknesses and strengths to competing products.

Acceptance Testing

Acceptance testing, which is black box testing, will give the client the opportunity to verify the system functionality and usability prior to the system being moved to production. The acceptance test will be the responsibility of the client, however, it will be conducted with full support from the project team. The Test Team will work with the client to develop the acceptance criteria.

Pointe Technology Group, Inc

Software Testing and Quality Assurance White Papers

Alpha Testing

Testing of an application when development is nearing completion; minor design changes may still be made as a result of such testing. Typically done by end-users or others, not by programmers or testers.

Beta Testing

Testing when development and testing are essentially completed and final bugs and problems need to be found before final release. Typically done by end-users or others, not by programmers or testers.

Resources

The following testing roles and skill sets are standard for most testing projects. Depending on the project, one or more of these roles are combined for one person. For instance a tester will also handle the role of Technical Analyst.

Test/QA Team Lead

- Coordinates the testing activity, communicates testing status to management and manages the test team.

Testers

- Develop test script/case and data, script execution, metrics analysis, and results evaluation for system, integration and regression testing.

Test Build Manager/System Administrator/Database Administrator

- Delivers current versions of software to the test environment.
- Performs installations of application software, applies patches (both application and operating system).
- Performs set-up, maintenance, and back up of test environment hardware.

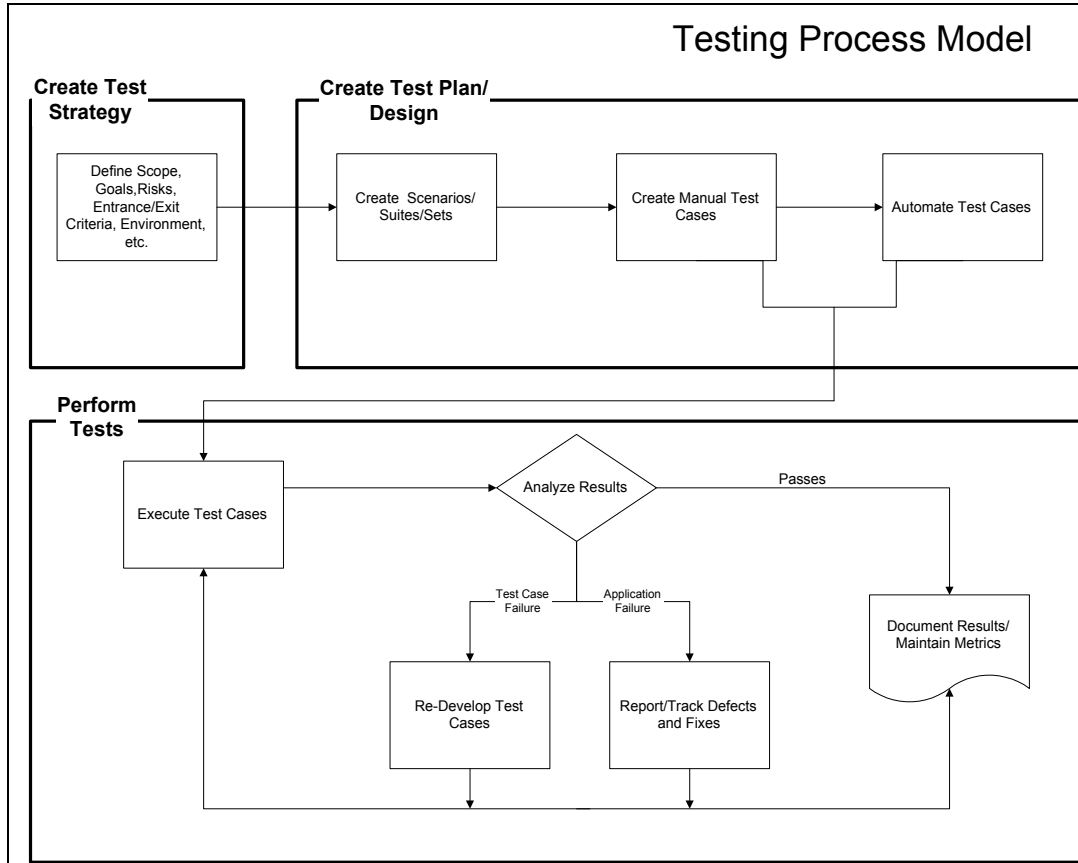
Technical Analyst & Test Configuration Manager

- Performs testing assessment and validates System/Functional test requirements.
- Maintains the test environment, scripts, software, and test data.

Software Testing Methodology

Pointe Technology can take their three step testing process will be employed and mold it to the organization's structure. We believe that using this methodology is important in development and in the ongoing maintenance of our customer applications.

Following is a graphic that depicts the general testing process.



For each testing level, as appropriate, the following activities will be performed:

Step 1 - Create Test Strategy

Inputs for this process:

- A description of the required hardware and software components including test tools (Test Environment, Test Tool Data).
- A description of roles and responsibilities of the resources required for the test and schedule constraints (Staff, Schedules).
- Testing methodology (Standards).
- Functional and technical requirements of the application (Requirements, Change Requests, Technical and Functional Design Documents).
- Requirements that the system can not provide (System Limitations)

Outputs for this process:

- An approved and signed-off test strategy document, test plan, test cases.
- Testing issues requiring resolution (usually requires the coordination of client project management).

Process:

- The test strategy is a formal description of how System XYZ will be tested. A test strategy will be developed for all levels of testing, as required. The Test Team will analyze the requirements, write the test strategy and review the plan with the project team.
- The test plan will include test cases and conditions, the testing environment, a list of testing related tasks, pass/fail criteria and testing risk assessment.. The test schedule will identify all tasks required for a successful testing effort, a schedule of activities, and resource requirements

Step 2 - Create Test Plan/Design

Inputs for this process:

- Approved Test Strategy Document.
- Automated testware and previously developed scripts, if applicable (Test Tools).
- Test document problems uncovered as a result of testing (Test Document Problems).
- Understanding of software complexity and module path coverage derived from General and Detailed Design documents (Software Design, Code, Complexity Data).

Outputs for this process:

- Problems with the design fed back to the developers (Software Design, Code Issues).

- Approved test scenarios, conditions and scripts (Test Design, Cases, Scripts).
- Test data.

Process:

- Test scenarios and cases will be prepared by reviewing the functional requirements of the release and preparing logical groups of business functions that can be further broken into test scripts. Tests will define test conditions, data to be used for testing, and expected results (database updates, file outputs, report results, etc.). Test scenarios will be designed to represent both typical and unusual situations that may occur in the application.
- The project developers will define the unit test requirements and unit test scenarios/cases. The developer will also be responsible for executing the unit test cases prior to the start of integration and system testing.
- Test scenarios/cases for Integration and System tests will be developed by the Test Team with assistance from developers and clients. Acceptance test cases will be developed by the client with help from the project and Test Team.
- Test scenarios will be executed through the use of test scripts. Scripts will define a series of steps necessary to perform one or more test scenarios. A test script usually represents a transaction or process that can occur during normal system operations. Test scripts will include the specific data that will be used for testing the process or transaction. Test scripts will cover multiple test scenarios and will include run/execution/cycle information. Test scripts will be mapped back to the requirements traceability matrices to ensure that each test is within scope.
- Test data will be captured and baselined prior to testing. This data will serve as the foundation for unit and system testing and will be used to exercise system functionality in a controlled environment. Some output data will also be baselined for future comparisons. Baselined data will also be used to support future application maintenance via regression testing.
- A pre-test meeting will be held to assess the “readiness” of the application, environment and data to be tested. A test readiness document will be created to indicate the status of the entrance criteria of the release.

Step 3 - Execute Tests

Inputs for this process:

- Approved test documents (Test Plan, Cases, Procedures)
- Automated testware, if applicable and developed scripts (Test Tools)
- Changes to the design (Change Requests)
- Test data
- Availability of the test and project teams (Project Staff, Test Team)
- General and Detailed Design Documents (Requirements, Software Design)
- A complete development environment that has been migrated to the test environment (Unit Tested Code) via the Configuration/Build Manager
- Test Readiness Document
- Update documents.

Outputs for this process:

- Changes to the code (Test Fixes)
- Test document problems uncovered as a result of testing (Test Document Problems)
- Problems with the design fed back to the developers and clients (Requirements, Design, Code Issues)
- Formal record of test incidents (Problem Tracking - PT)
- Baselined package ready for migration to the next level (Tested Source and Object Code)
- Log and summary of the test results (Test Report)
- Approved and signed-off revised testing deliverables (Updated Deliverables)

Process:

- Checkpoint meetings will be held throughout the Execution phase. The Checkpoint meeting will be held daily (if required) to address and discuss testing issues, status, and activities.
- Execution of tests is completed by following the test documents in a methodical manner. As each package of test procedures is performed, an entry is recorded in a test execution log to note the execution of the procedure and whether the test procedures uncovered any defects. The output from the execution of test procedures is referred to as test results.
- Test results will be evaluated by the appropriate project members, applicable to the level of test, to determine whether the expected results were obtained. All discrepancy/anomalies will be logged and discussed with the Software Development Manager/Programmer and documented for further investigation and resolution. Each client may have a different process for logging and reporting bugs/defects uncovered during testing, verify the process through the Configuration Management (CM) group. Pass/Fail criteria will be used to determine the severity of the problem. Results will be recorded in a test summary report.
- The severity of a problem found during system testing will be defined in accordance to the customer's risk assessment and recorded in their selected tracking tool. (Point is a Quality Channel Partner with Mercury Interactive. Mercury offers a test tool for each level of testing).
- Proposed fixes will be delivered to the testing environment based on the severity of the problem. Fixes will be regression tested and flawless fixes will be migrated to the new baseline. Following completion of the test, members of the Test Team will prepare a summary report. The summary report will be reviewed by the Project Manager, Clients, Software Quality Assurance (SQA) and/or Test Team Lead.
- After a particular level of testing has been certified, it will be the responsibility of Configuration Manager to coordinate the migration of the release software components to the next test level as documented in the Configuration Management Plan. The software will only be migrated to the production environment after the client has formally accepted it.
- The Test Team will review test document problems identified during the testing and update documents where appropriate. Some problems may be the result of inconsistencies or modifications between the Technical and Functional

Frequently Asked Questions

Why is it often hard for management to get serious about quality assurance?

Solving problems is a high-visibility process; preventing problems is low-visibility.

Why does software have bugs?

- Miscommunication or no communication - as to specifics of what an application should or shouldn't do (the application's requirements).
- Software complexity - the complexity of current software applications can be difficult to comprehend for anyone without experience in modern-day software development. Windows-type interfaces, client-server and distributed applications, data communications, enormous relational databases, and sheer size of applications have all contributed to the exponential growth in software/system complexity.
- Programming errors - programmers, like anyone else, can make mistakes.
- Changing requirements - the customer may not understand the effects of changes, or may understand and request them anyway - redesign, rescheduling of engineers, effects on other projects, work already completed that may have to be redone or thrown out, hardware requirements that may be affected, etc. If there are many minor changes or any major changes, known and unknown dependencies among parts of the project are likely to interact and cause problems, and the complexity of keeping track of changes may result in errors. Enthusiasm of engineering staff may be affected. In some fast-changing business environments, continuously modified requirements may be a fact of life. In this case, management must understand the resulting risks, and QA and test engineers must adapt and plan for continuous extensive testing to keep the inevitable bugs from running out of control - see 'What can be done if requirements are changing continuously?'
- Time pressures - scheduling of software projects is difficult at best, often requiring a lot of guesswork. When deadlines loom and the crunch comes, mistakes will be made.
- Poorly documented code - it's tough to maintain and modify code that is badly written or poorly documented; the result is bugs. In many organizations management provides no incentive for programmers to document their code or write clear, understandable code. In fact, it's usually the opposite: they get points mostly for quickly turning out code, and there's job security if nobody else can understand it ('if it was hard to write, it should be hard to read').
- Software development tools - visual tools, class libraries, compilers, scripting tools, etc. often introduce their own bugs or are poorly documented, resulting in added bugs.

How can new Software QA processes be introduced in an existing organization?

A lot depends on the size of the organization and the risks involved. For large organizations with high-risk (in terms of lives or money) projects, serious management buy-in is required and a formalized QA process is necessary.

Where the risk is lower, management and organizational buy-in and QA implementation may be a slower, step-at-a-time process. QA processes should be balanced with productivity so as to keep bureaucracy from getting out of hand.

For small groups or projects, a more ad-hoc process may be appropriate, depending on the type of customers and projects. A lot will depend on team leads or managers, feedback to developers, and ensuring adequate communications among customers, managers, developers, and testers.

In all cases the most value for effort will be in requirements management processes, with a goal of clear, complete, testable requirement specifications.

What is verification? validation?

Verification typically involves reviews and meetings to evaluate documents, plans, code, requirements, and specifications. This can be done with checklists, issues lists, walkthroughs, and inspection meetings.

Validation typically involves actual testing and takes place after verifications are completed.

What is a 'walkthrough'?

A 'walkthrough' is an informal meeting for evaluation or informational purposes.

What's an 'inspection'?

An inspection is more formalized than a 'walkthrough', typically with 3-8 people including a moderator, reader (the author of whatever is being reviewed), and a recorder to take notes. The subject of the inspection is typically a document such as a requirements or a test plan. The purpose is to find problems and see what's missing, not to fix anything.

Attendees should prepare for this type of meeting by reading thru the document; most problems will be found during this preparation. The result of the inspection meeting should be a written report. Preparation for inspections is difficult is one of the most cost effective methods of ensuring quality since bug prevention is far more cost effective than bug detection.

What are five common problems in the software development process?

- 1) Poor requirements - if requirements are unclear, incomplete, too general, or not testable, there will be problems.
- 2) Unrealistic schedule - if too much work is crammed in too little time, problems are inevitable.
- 3) Inadequate testing - no one will know whether or not the program is any good until the customer complains or systems crash.
- 4) Featuritic - requests to pile on new features after development is underway; extremely common.
- 5) Miscommunication - if developers don't know what's needed or customers have erroneous expectations, problems are guaranteed.

What are five common solutions to software development problems?

- 1) Solid requirements - clear, complete, detailed, cohesive, attainable, testable requirements that are agreed to by all players. Use prototypes to help nail down requirements.
- 2) Realistic schedules - allow adequate time for planning, design, testing, bug fixing, re-testing, changes, and documentation; personnel should be able to complete the project without burning out.
- 3) Adequate testing - start testing early on, re-test after fixes or changes, plan for adequate time for testing and bug fixing.
- 4) Stick to initial requirements as much as possible - be prepared to defend against changes and additions once development has begun, and be prepared to explain consequences. If changes are necessary, they should be adequately reflected in related schedule changes. If possible, use rapid prototyping during the design phase so that customers can see what to expect. This will provide them a higher comfort level with their requirements decisions and minimize changes later on.
- 5) Communication - require walkthroughs and inspections when appropriate; make extensive use of group communication tools - e-mail, groupware, networked bug-tracking tools and change management tools, intranet capabilities, etc.; insure that documentation is available and up-to-date - preferably electronic, not paper; promote teamwork and cooperation; use prototypes early on so that customers' expectations are clarified.

What is software 'quality'?

Quality software is reasonably bug-free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable. However, quality is obviously a subjective term. It will depend on who the 'customer' is and their overall influence in the scheme of things. A wide-angle view of the 'customers' of a software development project might include end-users, customer acceptance testers, customer contract officers, customer management, the development organization's management/accountants/testers/salespeople, future software maintenance engineers, stockholders, magazine columnists, etc. Each type of 'customer' will have their own slant on 'quality' - the accounting department might define quality in terms of profits while an end-user might define quality as user-friendly and bug-free.

What is 'good code'?

'Good code' is code that works, is bug free, and is readable and maintainable. Some organizations have coding 'standards' that all developers are supposed to adhere to, but everyone has different ideas about what's best, or what is too many or too few rules. There are also various theories and metrics. It should be kept in mind that excessive use of standards and rules can stifle productivity and creativity. 'Peer reviews', 'buddy checks' code analysis tools, etc. can be used to check for problems and enforce standards.

What is 'good design'?

'Design' could refer to many things, but often refers to 'functional design' or 'internal design'. Good functional design is indicated by an application whose functionality can be traced back to customer and end-user requirements. Good internal design is indicated by software code whose overall structure is clear, understandable, easily modifiable, and maintainable; is robust with sufficient error handling and status logging capability; and works correctly when implemented. (See further discussion of functional and internal design in 'What's the big deal about requirements?')

What is the 'software life cycle'?

The life cycle begins when an application is first conceived and ends when it is no longer in use. It includes aspects such as initial concept, requirements analysis, functional design, internal design, documentation planning, test planning, coding, document preparation, integration, testing, maintenance, updates, retesting, phase-out, and other aspects.

Will automated testing tools make testing easier?

Possibly. For small projects, the time needed to learn and implement them may not be worth it. For larger projects, or on-going long-term projects they can be valuable.

A common type of automated tool is the 'record/playback' type. For example, a tester could click through all combinations of menu choices, dialog box choices, buttons, etc. in an application GUI and have them 'recorded' and the results logged by a tool. The 'recording' is typically in the form of text based on a scripting language that is interpretable by the testing tool. If new buttons are added, or some underlying code in the application is changed, etc. the application can then be retested by just 'playing back' the 'recorded' actions, and comparing the logging results to check effects of the changes.

The problem with such tools is that if there are continual changes to the system being tested, the 'recordings' may have to be changed so much that it becomes very time-consuming to continuously update the scripts. Additionally, interpretation of results (screens, data, logs, etc.) can be a difficult task.

What makes a good test engineer?

Pointe's test engineers have a 'test to break' attitude, an ability to take the point of view of the customer, a strong desire for quality, and an attention to detail. Tact and diplomacy are useful in maintaining a cooperative relationship with developers, and an ability to communicate with both technical (developers) and non-technical (customers, management) people is useful. Pointe's previous software development experience is helpful as it provides a deeper understanding of the software development process, gives the tester an appreciation for the developers' point of view, and reduce the learning curve in automated test tool programming.

What makes a good Software QA engineer?

The same qualities a good tester has are useful for a QA engineer. Additionally, Pointe is able to understand the entire software development process and how it can fit into the business approach and goals of the organization. Communication skills and the ability to understand various sides of issues are important.

What makes a good QA or Test manager?

Pointe's QA or Test or QA/Test Managers are familiar with the software development process, able to maintain enthusiasm of their team and promote a positive atmosphere, able to promote teamwork to increase productivity, able to promote cooperation between software, test, and QA engineers, have the diplomatic skills needed to promote improvements in QA processes, have the ability to withstand pressures and say 'no' to other managers when quality is insufficient or QA processes are not being adhered to, able to communicate with technical and non-technical people, engineers, managers, and customers. As well as, able to run meetings and keep them focused.

What's the role of documentation in QA?

Documentation plays a critical role in QA. QA practices should be documented such that they are repeatable. Specifications, designs, business rules, inspection reports, configurations, code changes, test plans, test cases, bug reports, user manuals, etc. should all be documented. There should ideally be a system for easily finding and obtaining documents and determining what documentation will have a particular piece of information. Change management for documentation should be used if possible.

What's the big deal about 'requirements'?

One of the most reliable methods of insuring problems, or failure, in a complex software project is to have poorly documented requirements specifications. Requirements are the details describing an application's externally perceived functionality and properties. Requirements should be clear, complete, reasonably detailed, cohesive, attainable, and testable. A non-testable requirement would be, for example, 'user-friendly' (too subjective). A testable requirement would be something like 'the product shall allow the user to enter their previously-assigned password to access the application'.

Care should be taken to involve ALL of a project's significant 'customers' in the requirements process. 'Customers' could be in-house personnel or out, and could include end-users, customer acceptance testers, customer contract officers, customer management, future software maintenance engineers, salespeople, etc. Anyone who could later derail the project if his/her expectations aren't met should be included if possible.

In some organizations requirements may end up in high-level project plans, functional specification documents, in design documents, or in other documents at various levels of detail. No matter what they are called, some type of documentation with detailed requirements will be needed by testers in order to properly plan and execute tests. Without such documentation, there will be no clear-cut way to determine if a software application is performing correctly.

What's a 'test plan'?

A software project test plan is a document that describes the objectives, scope, approach, and focus of a software testing effort. The process of preparing a test plan is a useful way to think through the efforts needed to validate the acceptability of a software product. The completed document will help people outside the test group understand the 'why' and 'how' of product validation. It should be thorough enough to be useful but not so thorough that no one outside the test group will read it.

What's a 'test case'?

A test case is a document that describes an input, action, or event and expected result, to determine if a feature of an application is working correctly. A test case should contain particulars such as test case identifier, test case name, objective, test conditions/setup, input data requirements, steps, and expected results.

Note that the process of developing test cases can help find problems in the requirements or design of an application, since it requires completely thinking through the operation of the application. For this reason, it's useful to prepare test cases early in the development cycle if possible.

What should be done after a bug is found?

The bug needs to be communicated and assigned to developers that can fix it. After the problem is resolved, fixes should be re-tested, and determinations made regarding requirements for regression testing to check that fixes didn't create problems elsewhere. If a problem-tracking system is in place, it should encapsulate these processes. A variety of commercial problem-tracking/management software tools are available. These tools will give the team complete information such that developers can understand the bug, get an idea of it's severity, and reproduce it if necessary. (Pointe Technology is a partner with companies that provide test tools).

What is configuration management (CM)?

Configuration management covers the processes used to control, coordinate, and track: code, requirements, documentation, problems, change requests, designs, tools/compiler/libraries/patches, changes made to them, and who makes the changes. Pointe Technology has a full range of CM concepts that we can adapt to your software process needs.

What if the software is so buggy it can't really be tested at all?

The best bet in this situation is for the testers to go through the process of reporting whatever bugs or blocking-type problems initially show up, with the focus being on critical bugs. Since this type of problem can severely affect schedules, and indicates deeper problems in the software development process (such as insufficient unit testing or insufficient integration testing, poor design, improper build or release procedures, etc.) managers should be notified, and provided with some documentation as evidence of the problem.

How can it be known when to stop testing?

This can be difficult to determine. Many modern software applications are so complex, and run in such an interdependent environment, that complete testing can never be done. Common factors in deciding when to stop are:

- Deadlines (release deadlines, testing deadlines, etc.)
- Test cases completed with certain percentage passed
- Test budget depleted
- Coverage of code/functionality/requirements reaches a specified point
- Bug rate falls below a certain level
- Beta or alpha testing period ends

What if there isn't enough time for thorough testing?

Use risk analysis to determine where testing should be focused. Since it's rarely possible to test every possible aspect of an application, every possible combination of events, every dependency, or everything that could go wrong, risk analysis is appropriate to most software development projects. This requires judgment skills, common sense, and experience.

Considerations can include:

- Which functionality is most important to the project's intended purpose?
- Which functionality is most visible to the user?
- Which functionality has the largest safety impact?
- Which functionality has the largest financial impact on users?
- Which aspects of the application are most important to the customer?
- Which aspects of the application can be tested early in the development cycle?
- Which parts of the code are most complex, and thus most subject to errors?
- Which parts of the application were developed in rush or panic mode?
- Which aspects of similar/related previous projects caused problems?
- Which aspects of similar/related previous projects had large maintenance expenses?
- Which parts of the requirements and design is unclear or poorly thought out?
- What do the developers think are the highest-risk aspects of the application?
- What kinds of problems would cause the worst publicity?
- What kinds of problems would cause the most customer service complaints?
- What kinds of tests could easily cover multiple functionalities?
- Which tests will have the best high-risk-coverage to time-required ratio?

What if the project isn't big enough to justify extensive testing?

Consider the impact of project errors, not the size of the project. However, if extensive testing is still not justified, risk analysis is again needed and the same considerations as described previously in 'What if there isn't enough time for thorough testing?' apply. The tester might then do ad hoc testing, or write up a limited test plan based on the risk analysis.

What can be done if requirements are changing continuously?

Work with the project's stakeholders early on to understand how requirements might change so that alternate test plans and strategies can be worked out in advance, if possible.

It's helpful if the application's initial design allows for some adaptability so that later changes do not require redoing the application from scratch.

- If the code is well commented and well documented this makes changes easier for the developers.
- Use rapid prototyping whenever possible to help customers feel sure of their requirements and minimize changes.
- The project's initial schedule should allow for some extra time commensurate with the possibility of changes.
- Try to move new requirements to a 'Phase 2' version of an application, while using the original requirements for the 'Phase 1' version.
- Negotiate to allow only easily implemented new requirements into the project, while moving more difficult new requirements into future versions of the application.
- Be sure that customers and management understand the scheduling impacts, inherent risks, and costs of significant requirements changes. Then let management or the customers (not the developers or testers) decide if the changes are warranted - after all, that's their job.
- Balance the effort put into setting up automated testing with the expected effort required to re-do them to deal with changes.
- Try to design some flexibility into automated test scripts.
- Focus initial automated testing on application aspects that are most likely to remain unchanged.
- Devote appropriate effort to risk analysis of changes to minimize regression testing needs.
- Design some flexibility into test cases (this is not easily done; the best bet might be to minimize the detail in the test cases, or set up only higher-level generic-type test plans)
- Focus less on detailed test plans and test cases and more on ad hoc testing (with an understanding of the added risk that this entails).

What if the application has functionality that wasn't in the requirements?

It may take serious effort to determine if an application has significant unexpected or hidden functionality, and it would indicate deeper problems in the software development process. If the functionality isn't necessary to the purpose of the application, it should be removed, as it may have unknown impacts or dependencies that were not taken into account by the designer or the customer. If not removed, design information will be needed to determine added testing needs or regression testing needs. Management should be made aware of any significant added risks as a result of the unexpected functionality. If the functionality only effects areas such as minor improvements in the user interface, for example, it may not be a significant risk.

How can Software QA processes be implemented without stifling productivity?

By implementing QA processes slowly over time, using consensus to reach agreement on processes, and adjusting and experimenting as an organization grows and matures, productivity will be improved instead of stifled. Problem prevention will lessen the need for problem detection, panics and burnout will decrease, and there will be improved focus and less wasted effort. At the same time, attempts should be made to keep processes simple and efficient, minimize paperwork, promote computer-based processes and automated tracking and reporting, minimize time required in meetings, and promote training as part of the QA process. However, no one - especially talented technical types - likes rules or bureaucracy, and in the short run things may slow down a bit. A typical scenario would be that more days of planning and development will be needed, but less time will be required for late-night bug fixing and calming of irate customers.

What if organization is growing fast that fixed QA processes are impossible?

This is a common problem in the software industry, especially in new technology areas. There is no easy solution in this situation, other than:

- Hire good people (i.e.: Pointe Technology)
- Management should 'ruthlessly prioritize' quality issues and maintain focus on the customer
- Everyone in the organization should be clear on what 'quality' means to the customer

How does a client/server environment affect testing?

Client/server applications can be quite complex due to the multiple dependencies among clients, data communications, hardware, and servers. Thus testing requirements can be extensive. When time is limited (as it usually is) the focus should be on integration and system testing. Additionally, load/stress/performance testing may be useful in determining client/server application limitations and capabilities. There are commercial tools to assist with such testing which we can assist you in finding the right tool to meet your needs.

How can Web sites be tested?

Web sites are essentially client/server applications - with web servers and 'browser' clients. Consideration should be given to the interactions between HTML pages, protocols, security, applications that run in web pages (such as applets, JavaScript, plug-in applications), and applications that run on the server side (such as CGI scripts, database interfaces, logging applications, dynamic page generators, etc.). Additionally, there are a wide variety of servers and browsers, various versions of each, small but sometimes significant differences between them, variations in connection speeds, rapidly changing technologies, and multiple standards and protocols. The end result is that testing for web sites can become a major ongoing effort.

Pointe Technology can help you with your considerations for web testing that might include:

- What are the expected loads on the server (e.g., number of hits per unit time?), and what kind of performance is required under such loads (such as web server response time, database query response times).
- What kinds of tools will be needed for performance testing (such as web load testing tools, other tools already in house that can be adapted, web robot downloading tools, etc.)?
- Who is the target audience? What kind of browsers will they be using? What kinds of connection speeds will they be using? Are they intra-organization (thus with likely high connection speeds and similar browsers) or Internet-wide (thus with a wide variety of connection speeds and browser types)?
- What kind of performance is expected on the client side (e.g., how fast should pages appear, how fast should animations, applets, etc. load and run)?
- Will down time for server and content maintenance/upgrades be allowed?
- What kinds of security (firewalls, encryptions, passwords, etc.) will be required and what is it expected to do? How can it be tested?
- How reliable are the site's Internet connections required to be? And how does that affect backup system or redundant connection requirements and testing?
- What processes will be required to manage updates to the web site's content, and what are the requirements for maintaining, tracking, and controlling page content, graphics, links, etc.?
- Which HTML specification will be adhered to? How strictly? What variations will be allowed for targeted browsers?
- Will there be any standards or requirements for page appearance and/or graphics throughout a site or parts of a site??
- How will internal and external links be validated and updated? How often?
- Can testing be done on the production system, or will a separate test system be required? How are browser caching, variations in browser option settings, and real-world internet 'traffic congestion' problems to be accounted for in testing?
- How extensive or customized are the server logging and reporting requirements; are they considered an integral part of the system and do they require testing?

- How are CGI programs, applets, JavaScript, ActiveX components, etc. to be maintained, tracked, controlled, and tested?
- Pages should be 3-5 screens max unless content is tightly focused on a single topic. If larger, provide internal links within the page.
- The page layouts and design elements should be consistent throughout a site, so that it's clear to the user that they're still within a site.
- Pages should be as browser-independent as possible, or pages should be provided or generated based on the browser-type.
- All pages should have links external to the page; there should be no dead-end pages.
- The page owner, revision date, and a link to a contact person or organization should be included on each page.

For more information see Pointe Technology's *White Papers on Website Testing*.

How is testing affected by object-oriented designs?

Well-engineered object-oriented design can make it easier to trace from code to internal design to functional design to requirements. While there will be little affect on black box testing (where an understanding of the internal design of the application is unnecessary), white-box testing can be oriented to the application's objects. If the application was well designed this can simplify test design.

Why is it recommended to test during the design phase?

Testing during the design phase can prevent defects later. We recommend verifying three things:

- 1) Verify design is good – efficient, compact, testable, maintainable product.
- 2) Verify design meets the requirements and is complete (specifies all relationships between the modules, how pass data, what happens in exceptional circumstances, starting state of each module, and how guarantee the modules state).
- 3) Verify the design incorporates enough memory, I/O devices, and quick enough runtime for the final product.

What are the five approaches to integration testing?

Pointe Technology can perform integration testing one of five approaches.

- 1) Bottom-Up Approach: Begins the test with lower-level modules or subsystems and progresses upward to the main program/module or subsystem. A structure chart is necessary to determine the order of execution. The development of drivers is necessary to complete the bottom-up approach.
- 2) Top-Down Approach: Starts testing with the main module/subsystem and progresses down lower-level modules/subsystems.
- 3) Data Flow Approach: Focuses on the integration of programs/modules/subsystems by the flow of data.
- 4) Capacity Approach: Tests in terms of functional groups, not using the structured analysis the above three approach uses.
- 5) Non-incremental or "Big Bang" Approach: Combines all programs/modules/subsystems for execution at once.

What are the three software development process models?

- 1) Waterfall: linear progression of project activities
- 2) Spiral: bends planning, requirements and design of waterfall back around three times to allow these activities to be injected with activities of evolution, risk, verification, and planning based on results of previous spiral. When get to implementation activities follow waterfall model.

Evolutionary: output from each development activity is fed both back and forward.

Glossary of Terms

A

Acceptable quality level (AQL)

Quality level accepted by all the stakeholders.

Acceptance sampling

Inspection of a sample from a lot to decide whether to accept or not accept that lot. There are two types: attributes sampling and variables sampling. In attributes sampling, the presence or absence of a characteristic is noted in each of the units inspected. In variables sampling, the numerical magnitude of a characteristic is measured and recorded for each inspected unit; this involves reference to a continuous scale of some kind.

Acceptance sampling plan

A specific plan that indicates the sampling sizes and the associated acceptance or no acceptance criteria to be used in testing. In attributes sampling, for example, there are single, double, multiple, sequential, chain, and skip-lot sampling plans. In variables sampling, there are single, double, and sequential sampling plans.

ANSI

American National Standards Institute

ASQ

A society of individual and organizational members dedicated to the ongoing development, advancement, and promotion of quality concepts, principles, and technologies. The Society serves more than 130,000 individuals and 1000 corporate members in the United States and 63 other countries. (Some of Pointe Technology's consultants are members).

Availability

The ability of a product to be in a state to perform its designated function under stated conditions at a given time.

Average outgoing quality (AOQ)

The expected average quality level of outgoing product for a given value of incoming product quality.

Average outgoing quality limit (AOQL)

The maximum average outgoing quality over all possible levels of incoming quality for a given acceptance sampling plan and disposal specification.

B

Baseline

A specification or product that has been formally reviewed and agreed upon, that thereafter serves as the basis for further development, and that can be changed only through formal change control procedures.

Benchmarking

An improvement process in which a company measures its performance against that of best-in-class companies, determines how those companies achieved their performance levels, and uses the information to improve its own performance. The subjects that can be benchmarked include strategies, operations, processes, and procedures. (Similar to nominal group technique.)

Big Q, little Q

A term used to contrast the difference between managing for quality in all business processes and products (big Q) and managing for quality in a limited capacity – traditionally in only factory products and processes (little q).

Blemish

An imperfection that is severe enough to be noticed but should not cause any real impairment with respect to intended normal or reasonably foreseeable use. Also see “defect,” “imperfection,” and “nonconformity.”

Block diagram

A diagram that shows the operation, interrelationships, and interdependencies of components in a system. Boxes, or blocks (hence the name), represent the components; connecting lines between the blocks represent interfaces.

There are two types of block diagrams:

functional block diagram: shows a system’s subsystems and lower-level products, their interrelationships, and interfaces with other systems

reliability block diagram: similar to the functional block diagram except that it is modified to emphasize those aspects influencing reliability.

C

Calibration

The comparison of a measurement instrument or system of unverified accuracy to a measurement instrument or system of a known accuracy to detect any variation from the required performance specification. (Parallel Testing)

Cause-and-effect diagram

A tool for analyzing process dispersion. It is also referred to as the Ishikawa diagram, because Kaoru Ishikawa developed it, and the fishbone diagram, because the complete diagram resembles a fish skeleton. The diagram illustrates the main causes and subclasses leading to an effect (symptom). The cause-and-effect diagram is one of the seven tools of quality.

Check sheet

A simple data-recording device. The check sheet is custom-designed by the user, which allows him or her to readily interpret the results. The check sheet is one of the seven tools of

quality. Check sheets are often confused with data sheets and checklists (see individual entries).

Checklist

A tool used to ensure that all important steps or actions in an operation have been taken. Checklists contain items that are important or relevant to an issue or situation. Checklists are often confused with check sheets and data sheets (see individual entries).

Common causes

Causes of variation that is inherent in a process over time. They affect every outcome of the process and everyone working in the process (see also “special causes”).

Company culture

A system of values, beliefs, and behaviors inherent in a company. To optimize business performance, top management must define and create the necessary culture.

Conformance

An affirmative indication or judgment that a product or service has met the requirements of a relevant specification, contract, or regulation.

Continuous improvement

The ongoing improvement of products, services, or processes through incremental and breakthrough improvements.

Control chart

A chart with upper and lower control limits on which values of some statistical measure for a series of samples or subgroups are plotted. The chart frequently shows a central line to help detect a trend of plotted values toward either control limit. (Can use to plot found defects against the systems components).

Corrective action

The implementation of solutions resulting in the reduction or elimination of an identified problem.

Cost of poor quality

The costs associated with providing poor-quality products or services. There are four categories of costs:

Internal failure costs: costs associated with defects found before the customer receives the product or service

External failure costs: costs associated with defects found after the customer receives the product or service

Appraisal costs: costs incurred to determine the degree of conformance to quality requirements

Prevention costs: costs incurred to keep failure and appraisal costs to a minimum).

Cost of quality (COQ)

A term coined by Philip Crosby referring to the cost of poor quality.

Count chart

A control chart for evaluating the stability of a process in terms of the count of events of a given classification occurring in a sample. (Can use to plot out defects against this chart.)

Count-per-unit chart

A control chart for evaluating the stability of a process in terms of the average count of events of a given classification per unit occurring in a sample.

Crosby, Philip

The founder and chairman of the board of Career IV, an executive management consulting firm. Crosby also founded Philip Crosby Associates, Inc. and the Quality College. He has written many books, including *Quality Is Free*, *Quality Without Tears*, *Let's Talk Quality*, and *Leading: The Art of Becoming an Executive*. Crosby, who originated the zero defects concept, is an ASQ senior member and past president.

Customer

See "external customer" and "internal customer"

Customer delight

The result of delivering a product or service that exceeds customer expectations.

Customer satisfaction

The result of delivering a product or service that meets customer requirements.

Customer supplier partnership

A long-term relationship between a buyer and supplier characterized by teamwork and mutual confidence. The supplier is considered an extension of the buyer's organization. The partnership is based on several commitments. The buyer provides long-term contracts and uses fewer suppliers. The supplier implements quality assurance processes so that incoming inspection can be minimized. The supplier also helps the buyer reduce costs and improve product and process designs.

D

Decision matrix

A matrix used by teams to evaluate problems or possible solutions. After a matrix is drawn to evaluate possible solutions, for example, the team lists them in the far-left vertical column. Next, the team selects criteria to rate the possible solutions, writing them across the top row. Third, each possible solution is rated on a scale of 1 to 5 for each criterion and the rating recorded in the corresponding grid. Finally, the ratings of all the criteria for each possible solution are added to determine its total score. The total score is then used to help decide which solution deserves the most attention.

Defect

A product's or service's no fulfillment of an intended requirement or reasonable expectation for use, including safety considerations.

Demerit chart

A control chart for evaluating a process in terms of a demerit (defect), i.e., a weighted sum of counts of various classified nonconformities.

Deming Cycle

See “plan-do-check-act cycle”.

Deming Prize

Award given annually to organizations that, according to the award guidelines, have successfully applied company wide quality control based on statistical quality control and will keep up with it in the future. Although the award is named in honor of W. Edwards Deming, its criteria are not specifically related to Deming’s teachings.

Deming, W. Edwards

A prominent consultant, teacher, and author on the subject of quality. After sharing his expertise in statistical quality control to help the U.S. war effort during World War II, the War Department sent Deming to Japan in 1946 to help that nation recover from its wartime losses. Deming published more than 200 works, including the well-known books *Quality, Productivity, and Competitive Position* and *Out of the Crisis*. Deming developed the 14 points for managing. (See Fourteen points).

Dependability

The degree to which a product is operable and capable of performing its required function at any randomly chosen time during its specified operating time, provided that the product is available at the start of that period. (No operation-related influences are included.)

Dependability can be expressed by the ratio: time available divided by (time available + time required)

Design of experiments (DOE)

A branch of applied statistics dealing with planning, conducting, analyzing, and interpreting controlled tests to evaluate the factors that control the value of a parameter or group of parameters.

Designing in quality vs. inspecting in quality

See “prevention vs. detection”

Diagnostic journey and remedial journey

A two-phase investigation used by teams to solve chronic quality problems. In the first phase, the diagnostic journey, the team journeys from the symptom of a chronic problem to its cause. In the second phase, the remedial journey, the team journeys from the cause to its remedy.

E

80-20

Pointe Technology Group, Inc

Software Testing and Quality Assurance White Papers

A term referring to the Pareto principle, which was first defined by J. M. Juran in 1950. The principle suggests that most effects come from relatively few causes, that is, 80% of the effects come from 20% of the possible causes.

Employee involvement

A practice within an organization whereby employees regularly participate in making decisions on how their work areas operate, including making suggestions for improvement, planning, monitoring performance, and goal setting.

Empowerment

A condition whereby employees have the authority to make decisions and take action in their work areas without prior approval.

Experimental design

A formal plan that details the specifics for conducting an experiment, such as which responses, factors, levels, blocks, treatments, and tools are to be used.

External customer

A person or organization that receives a product, a service, or information but is not part of the organization supplying it. (See also "internal customer.")

F

Failure mode analysis (FMA)

A procedure to determine which malfunction symptoms appears immediately before or after a failure of a critical parameter in a system. After all the possible causes are listed for each symptom, the product is designed to eliminate the problems.

Failure mode effects analysis (FMEA)

A procedure in which each potential failure mode in every sub-item of an item is analyzed to determine its effect on other sub-items and on the required function of the item.

Failure mode effects and criticality analysis (FMECA)

A procedure that is performed after a failure mode effects analysis to classify each potential failure effect according to its severity and probability of occurrence

Featuristic

Requests to pile on new features after development are underway; extremely common.

Fitness for use

A term used to indicate that a product or service fits the customer's defined purpose for that product or service.

Flowchart

A graphical representation of the steps in a process. Flowcharts are drawn to better understand processes. The flowchart is one of the seven tools of quality.

Force field analysis

A technique for analyzing the forces that aid or hinder an organization in reaching an objective. An arrow pointing to an objective is drawn down the middle of a piece of paper.

Pointe Technology Group, Inc

Software Testing and Quality Assurance White Papers

The factors that will aid the objective's achievement, called the driving forces, are listed on the left side of the arrow. The factors that will hinder its achievement, called the restraining forces, are listed on the right side of the arrow.

Fourteen points

W. Edward Deming's 14 management practices to help companies increase their quality and productivity:

1. Create constancy of purpose for improving products and services,
2. Adopt the new philosophy,
3. Cease dependence on inspection to achieve quality,
4. End the practice of awarding business on price alone; instead, minimize total cost by working with a single supplier,
5. Improve constantly and forever every process for planning, production, and service,
6. Institute training on the job,
7. Adopt and institute leadership,
8. Drive out fear,
9. Break down barriers between staff areas,
10. Eliminate slogans, exhortations, and targets for the work force,
11. Eliminate numerical quotas for the work force and numerical goals for management,
12. Remove barriers that rob people of pride of workmanship and eliminate the annual rating or merit system,
13. Institute a vigorous program of education and self-improvement for everyone and,
14. Put everybody in the company to work to accomplish the transformation.

G

Gantt chart

A type of bar chart used in process planning and control to display planned work and finished work in relation to time.

Go/no-go

State of a unit or product.

Two possible parameters:

Go: conforms to specifications, and

No-go: does not conform to specifications.

H

Histogram

A graphic summary of variation in a set of data. The pictorial nature of the histogram lets people see patterns that are difficult to see in a simple table of numbers. The histogram is one of the seven tools of quality.

hoshin planning

Breakthrough planning. A Japanese strategic planning process in which a company develops up to four vision statements that indicate where the company should be in the next five years. Company goals and work plans are developed based on the vision statements. Periodic audits are then conducted to monitor progress.

I

IEEE

Institute of Electrical and Electronics Engineers

Imperfection

A quality characteristic's departure from its intended level or state without any association to conformance to specification requirements or to the usability of a product or service (see also "blemish," "defect," and "nonconformity").

In-control process

A process in which the statistical measure being evaluated is in a state of statistical control, i.e., the variations among the observed sampling results can be attributed to a constant system of chance causes (see also "out-of-control process").

Inspection

Measuring, examining, testing, and gauging one or more characteristics of a product or service and comparing the results with specified requirements to determine whether conformity is achieved for each characteristic.

Instant pudding

A term used to illustrate an obstacle to achieving quality: the supposition that quality and productivity improvement is achieved quickly through an affirmation of faith rather than through sufficient effort and education. W. Edwards Deming used this term, which was initially coined by James Bakken of Ford Motor Co., in his book *Out of the Crisis*.

Internal customer

The recipient, person or department, of another person's or department's output (product, service, or information) within an organization (see also "external customer").

ISO

International Organization for Standardization

ISO 9000 series standards

A set of five individual but related international standards on quality management and quality assurance developed to help companies effectively document the quality system elements to be implemented to maintain an efficient quality system. The standards, initially published in 1987, are not specific to any particular industry, product, or service. The International Organization developed the standards for Standardization (ISO), a specialized international agency for standardization composed of the national standards bodies of 91 countries.

J

Just-in-time manufacturing (JIT)

An optimal material requirement planning system for a manufacturing process in which there is little or no manufacturing material inventory on hand at the manufacturing site and little or no incoming inspection.

K

kaizen

A Japanese term that means gradual unending improvement by doing little things better and setting and achieving increasingly higher standards. Masaaki Imai made the term famous in his book, *Kaizen: The Key to Japan's Competitive Success*.

L

Leadership

An essential part of a quality improvement effort. Organization leaders must establish a vision, communicate that vision to those in the organization, and provide the tools and knowledge necessary to accomplish the vision.

Lot

A defined quantity of product accumulated under conditions that are considered uniform for sampling purposes.

Lower control limit (LCL)

Control limit for points below the central line in a control chart.

M

Maintainability

The probability that a given maintenance action for an item under given usage conditions can be performed within a stated time interval when the maintenance is performed under stated conditions using stated procedures and resources. Maintainability has two categories:

serviceability: the ease of conducting scheduled inspections and servicing

reparability: the ease of restoring service after a failure.

Malcolm Baldrige National Quality Award (MBNQA)

An award established by Congress in 1987 to raise awareness of quality management and to recognize U.S. companies that has implemented successful quality management systems.

Two awards may be given annually in each of three categories: manufacturing company, service company, and small business. The award is named after the late Secretary of Commerce Malcolm Baldrige, a proponent of quality management. The U.S. Commerce Department's National Institute of Standards and Technology manages the award, and ASQ administers it.

Mean time between failures (MTBF)

The average time interval between failures for repairable product for a defined unit of measure, for example, operating hours, cycles, miles.

MIL-STD

Military standard.

Multivariate control chart

A control chart for evaluating the stability of a process in terms of the levels of two or more variables or characteristics.

N

n

Sample size (the number of units in a sample)

NIST

National Institute of Standards and Technology

Nominal group technique

A technique similar to brainstorming, used by teams to generate ideas on a particular subject. Team members are asked to silently come up with as many ideas as possible, writing them down. Each member is then asked to share one idea, which is recorded. After all the ideas are recorded, they are discussed and prioritized by the group.

Nonconformity

The no fulfillment of a specified requirement (see also “blemish,” “defect,” and “imperfection”).

Nondestructive testing and evaluation (NDT)

Testing and evaluation methods that do not damage or destroy the product being tested.

NQM

Pointe Technology celebrates National Quality Month.

O

Out-of-control process

A process in which the statistical measure being evaluated is not in a state of statistical control, i.e., the variations among the observed sampling results can be attributed to a constant system of chance causes (see also “in-control process”).

Out of spec

A term used to indicate that a unit does not meet a given specification.

P

Pareto chart

A graphical tool for ranking causes from most significant to least significant. It is based on the Pareto principle, which was first defined by J. M. Juran in 1950. The principle, named after 19th-century economist Vilfredo Pareto, suggests that most effects come from relatively few causes; that is, 80% of the effects come from 20% of the possible causes. The Pareto chart is one of the seven tools of quality.

Percent chart

A control chart for evaluating the stability of a process in terms of the percent of the total number of units in a sample in which an event of a given classification occurs. The percent chart is also referred to as a proportion chart.

Plan-do-check-act cycle (PDCA)

A four-step process for quality improvement.

Plan: a plan to effect improvement is developed.

Do: the plan is carried out, preferably on a small scale.

Check: the effects of the plan are observed.

Pointe Technology Group, Inc

Software Testing and Quality Assurance White Papers

Act: the results are studied to determine what was learned and what can be predicted. The plan-do-check-act cycle is sometimes referred to as the Shewhart cycle because Walter Shewhart discussed the concept in his book *Statistical Method From the Viewpoint of Quality Control* and as the Deming cycle because W. Edwards Deming introduced the concept in Japan. The Japanese subsequently called it the Deming cycle.

Prevention vs. detection

A term used to contrast two types of quality activities. Prevention refers to those activities designed to prevent nonconformance in products and services. Detection refers to those activities designed to detect nonconformance already in products and services. Another term used to describe this distinction is “designing in quality vs. inspecting in quality.”

Product or service liability

The obligation of a company to make restitution for loss related to personal injury, property damage, or other harm caused by its product or service.

Q

QA

Quality assurance

QC

Quality Control

QIC

Quality Information Center

QMJ

Quality Management Journal

QP

Quality Progress

Quality

A subjective term for which each person has his or her own definition.

In technical usage quality can have two meanings:

The characteristics of a product or service that bear on its ability to satisfy stated or implied needs and a product or service free of deficiencies.

Quality assurance/quality control

Often, however, “quality assurance” and “quality control” are used interchangeably, referring to the actions performed to ensure the quality of a product, service, or process.

Quality assurance is all the planned and systematic activities implemented within the quality system that can be demonstrated to provide confidence that a product or service will fulfill requirements for quality.

Quality control is the operational techniques and activities used to fulfill requirements for quality.

Quality audit

A systematic, independent examination and review to determine whether quality activities and related results comply with planned arrangements and whether these arrangements are implemented effectively and are suitable to achieve the objectives.

Quality circles

Quality improvement or self-improvement study groups composed of a small number of employees – 10 or fewer – and their supervisor. Quality circles originated in Japan, where they are called quality control circles.

Quality control (QC)

See “quality assurance/quality control”

Quality costs

See “cost of poor quality”

Quality engineering

The analysis of a manufacturing system at all stages to maximize the quality of the process itself and the products it produces.

Quality function deployment (QFD)

A structured method in which customer requirements are translated into appropriate technical requirements for each stage of product development and production. The QFD process is often referred to as listening to the voice of the customer.

Quality loss function

A parabolic approximation of the quality loss that occurs when a quality characteristic deviates from its target value. The quality loss function is expressed in monetary units: the cost of deviating from the target increases quadratically the farther the quality characteristic moves from the target. The formula used to compute the quality loss function depends on the type of quality characteristic being used. Genichi Taguchi first introduced the quality loss function in this form.

Quality score chart (Q chart)

A control chart for evaluating the stability of a process in terms of a quality score. The quality score is the weighted sum of the count of events of various classifications where each classification is assigned a weight.

Quality trilogy

A three-pronged approach to managing for quality.

The three legs are

- Quality planning (developing the products and processes required to meet customer needs),
- Quality control (meeting product and process goals), and
- Quality improvement (achieving unprecedented levels of performance).

R

RAM

Reliability/availability/maintainability (see individual entries).

Random sampling

A commonly used sampling technique in which sample units are selected in such a manner that all combinations of n units under consideration have an equal chance of being selected as the sample.

Reliability

The probability of a product performing its intended function under stated conditions without failure for a given period of time.

Right the first time

A term used to convey the concept that it is beneficial and more cost-effective to take the necessary steps up front to ensure a product or service meets its requirements than to provide a product or service that will need rework or not meet customers' needs. In other words, an organization should engage in defect prevention rather than defect detection.

Robustness

The condition of a product or process design that remains relatively stable with a minimum of variation even though factors that influence operations or usage, such as environment and wear, are constantly changing.

S

Sample standard deviation chart (s chart)

A control chart in which the subgroup standard deviation, s, is used to evaluate the stability of the variability within a process.

Scatter diagram

A graphical technique to analyze the relationship between two variables. Two sets of data are plotted on a graph, with the y-axis being used for the variable to be predicted and the x-axis being used for the variable to make the prediction. The graph will show possible relationships (although two variables might appear to be related, they might not be: those who know most about the variables must make that evaluation). The scatter diagram is one of the seven tools of quality.

Seven tools of quality for ASQ

Tools that help organizations understand their processes in order to improve them.

1. Cause-and-effect diagram
2. Check sheet
3. Control chart
4. Flowchart
5. Histogram

6. Pareto chart
7. Scatter diagram (see individual entries).

Special causes

Causes of variation that arise because of special circumstances. They are not an inherent part of a process. Special causes are also referred to as assignable causes (see also “common causes”).

Specification

A document that states the requirements to which a given product or service must conform.

Statistical process control (SPC)

The application of statistical techniques to control a process. Often the term “statistical quality control” is used interchangeably with “statistical process control.”

Statistical quality control (SQC)

The application of statistical techniques to control quality. Often the term “statistical process control” is used interchangeably with “statistical quality control,” although statistical quality control includes acceptance sampling as well as statistical process control.

Supplier quality assurance

Confidence that a supplier’s product or service will fulfill its customers’ needs. This confidence is achieved by creating a relationship between the customer and supplier that ensures the product will be fit for use with minimal corrective action and inspection.

According to J. M. Juran, there are nine primary activities needed:

1. define product and program quality requirements,
2. evaluate alternative suppliers,
3. select suppliers,
4. conduct joint quality planning,
5. cooperate with the supplier during the execution of the contract,
6. obtain proof of conformance to requirements,
7. certify qualified suppliers,
8. conduct quality improvement programs as required,
9. create and use supplier quality ratings.

T

Tampering

Action taken to compensate for variation within the control limits of a stable system.

Tampering increases rather than decreases variation, as evidenced in the funnel experiment.

Top-management commitment

Participation of the highest-level officials in their organization’s quality improvement efforts. Their participation includes establishing and serving on a quality committee, establishing quality policies and goals, deploying those goals to lower levels of the organization, providing the resources and training that the lower levels need to achieve the goals, participating in quality improvement teams, reviewing progress organization wide; recognizing those who have performed well, and revising the current reward system to reflect the importance of achieving the quality goals.

Total quality management (TQM)

TQM is a management approach to long-term success through customer satisfaction. TQM is based on the participation of all members of an organization in improving processes, products, services, and the culture they work in. TQM benefits all organization members and society. The methods for implementing this approach are found in the teachings of such quality leaders as Philip B. Crosby, W. Edwards Deming, Armand V. Feigenbaum, Kaoru Ishikawa, and J. M. Juran.

Trend control chart

A control chart in which the deviation of the subgroup average, \bar{X} , from an expected trend in the process level is used to evaluate the stability of a process.

Type I error

An incorrect decision to reject something when it is acceptable.

Type II error

An incorrect decision to accept something when it is unacceptable.

U

Upper control limit (UCL)

Control limit for points above the central line in a control chart.

V

Value-adding process

Those activities that transform an input into a customer-usable output. The customer can be internal or external to the organization.

Variation

A change in data, a characteristic, or a function that is caused by one of four factors: special causes, common causes, tampering, or structural variation (see individual entries).

W

World-class quality

A term used to indicate a standard of excellence: best of the best.

Z

Zero defects

A performance standard developed by Philip B. Crosby to address a dual attitude in the workplace: people are willing to accept imperfection in some areas, while, in other areas; they expect the number of defects to be zero. This dual attitude had developed because of the conditioning that people are human and humans make mistakes. However, the zero defects methodology states that, if people commit themselves to watching details and avoiding errors, they can move closer to the goal of zero.

Pointe Technology Group, Inc

Software Testing and Quality Assurance White Papers

Contact

For more information on how Pointe Technology can provide quality assurance and/or software testing to your project please call Mr. John Zadjura

Mr. John Zadjura
Vice President Sales
8201 Corporate Drive, Suite 700
Landover, MD 20785
jzadjura@pointetech.com
301/306-4400, extension 4418

Bibliography

"Software Testing White Papers, An Introduction to Software Testing" Quality Checked Software, IPL Information Processing Ltd.,
Available <http://www.teleport.com/~qcs/papers/p820.htm>.

Hover, Rick. *Software QA and Testing Frequently-Asked-Questions (Parts 1 and 2)*
Available <http://www.softwareqatest.com/about.html>.

ASQ Glossary of Terms. American Society of Quality.
Available <http://www.asq.org/abtquality/definition.html>.