

Building a Quality Assurance (QA) and Testing Organization from Scratch

Hilary Benoit

W R Systems, Ltd.

10680 Main Street, Suite 300

Fairfax, VA 22030-3806

Phone: (703) 934 - 0200

Fax: (703) 934 - 0202

Email: hbenoit@wrsystems.com

Abstract

The ability to design, develop, and deliver a quality software product on time and within budget is, of course, the goal of every software development organization. The importance of Quality Assurance (QA) and testing is supported by such initiatives as the Carnegie Mellon Software Engineering Institute's (SEI) Capability Maturity Model (CMM) which describes a framework for a software development process that firmly integrates quality assurance and testing into the software development life cycle. By "building in" the quality from the very beginning of the software development life cycle, and by emphasizing the need for static and dynamic test methods throughout the life cycle, the cost of rework can be significantly reduced.

Building a QA & Test organization from scratch is not an easy task. It invariably involves breaking down a lot of barriers and overcoming multiple problems such as: misconceptions as to the importance of testing or of having trained personnel doing the testing, resistance from the project managers regarding having "extra" people depleting their budgets, prejudice concerning the "importance" or technical ability of non-developers, training management, training the testers, etc.

This paper documents the real-life experience of creating a QA & Test organization from scratch in a small, but growing, software engineering company. It describes a step-by-step process for building an effective software QA and Test function throughout the organization that will become an integral part of the software development process. Real-life examples and lessons learned illustrate what worked and what didn't.

A note on the difference between Quality Assurance and Testing

The Quality Assurance function is not the same as testing. Testing is generally considered to be a Quality Control activity, a means of "testing in" quality at the end of the life cycle rather than "building in" quality throughout the life cycle. However, when you are building your QA and Testing organization, you'll probably find that the distinctions, unhappily, become blurred to all of the players: the project managers, the Customer, the developers. Most small to mid-size companies are either unable or unwilling to come up with the resources required to establish two separate functions. It is therefore important that the people you build the QA and Test function with are able to cover both camps while understanding the difference. In our case study, we concentrated at first on the QA function and process, and gradually extended the organization and activities to include more of the test process.

Step 1 – Establish the need for QA and Testing

Building a QA and Test Organization from Scratch

Benoit

There are some alarming statistics about these days concerning software development. In general, 40% of software costs are spent on debugging – defect removal through inspections walkthroughs, audit, and testing. It typically costs three times as much to debug a program as it did to write it. In addition, an estimated 60% of all enhancements correct specification and requirements bugs. The cost of poor quality – not doing it right the first time – in the software industry today is an estimated \$100 billion.

Apparently, some 75% of software projects undertaken are never completed. We did not want to become part of those statistics! All too often, testing and QA are seen as trivial activities that happen at the end of the development life cycle to ensure that "everything works" prior to customer delivery. Prior to our efforts to establish an independent QA and Testing organization, we had in place a documented software development process but any reviews and testing were carried out by the development staff. QA activities tended to be mainly in the area of proofreading documentation. On our case-study project, the customer required there to be a full-time independent QA and testing function which would be responsible for developing and undertaking QA and testing activities throughout the life cycle. This seemed to be an excellent time to develop a QA and Testing organization from scratch.

Step 2 – Get Upper Management Support

The support of upper management is essential when trying to establish a QA and Testing organization. Any effort to establish a serious quality assurance and testing process in an organization will undoubtedly fail without the solid support of upper management. There may be a need to educate upper management in the cost benefits of establishing and maintaining an independent organization for QA and testing. Presentations on such topics as the Cost of Quality, or the Carnegie Mellon Software Engineering Institute's Capability Maturity Model (SEI CMM) may help to establish the benefits of such an organization. QA/Testers are often perceived as "dead weight" unless they are actually in the throes of heavy testing just prior to delivery of the software, and it may be a challenge to convince management that a permanent organization is worthwhile. The cost of Quality Assurance and Testing should ideally be included in the estimated project costs. The general recommendation is that it should range, at a very minimum, from 2.5% and 5% of the total project cost. Actual dynamic testing phases will normally require additional budget resources and can be expensive. Another school of thought is that QA and Testing combined should take up at least 20% of the project cost. Our QA specialist on our case study project was full-time and directly billable to the project.

Step 3 - Define and Document the Software Development Process

As a foundation to the QA and Test process, you need to start with a well-defined Software Development process with clear life cycle phases, and entry and exit criteria, in order to establish QA procedures appropriate to each phase. It doesn't have to be complicated, but it does have to be clear, workable, and logical. If you want or need the added bonus of compliance with the Carnegie Mellon Software Engineering Institute's Capability Maturity Model (SEI CMM), you should ensure that your software process includes all of the appropriate Key Practice Areas (KPA's).

Since the case study project was being developed for a major U.S. Government agency, and CMM level 2 compliance was a requirement of the contract, we needed to include specific activities into our QA process. WR Systems defined and documented a comprehensive full life-cycle software development

Building a QA and Test Organization from Scratch

Benoit

process with well-defined life-cycle phases. Documented entry and exit criteria, and quality checkpoints and reviews meant that QA would be essentially built into the process and that the QA and Testing organization would have a good starting point for its activities. Our defined life cycle phases were:

Strategy	Analysis	Design	Build and Test	Deployment	Maintenance
----------	----------	--------	----------------	------------	-------------

Step 4 - Set up the QA and Test Function

Once the need for a QA and Test function has been established, and management has given its blessing, the question is: how do you staff it? Even if you, as we did, have to start with only one person spearheading the initiative, it's important to get the right kind of person, since the person selected for the pilot project may well be required to manage future QA and Test activities and to grow the QA and Test organization. QA and testing is not a trivial job. The right attitude is extremely important. Ideally, you need someone with a solid background in software development who wants to get into the area of QA and/or testing. The selected person needs to be a good communicator and someone who will literally "carry the flag" for the cause against, quite often, less than responsive co-workers. The QA specialist on our large test project came with 15-year background in software engineering and saw moving into Quality Assurance as both a challenge and the opportunity to learn a new area of software engineering. The QA position on the project was full-time. This was important since the QA specialist was in a position to dedicate considerable time and energy to nothing but QA and Testing. The QA specialist initially reported directly to executive management which enabled us to establish the QA function as independent of the development organization. This was important in order to prevent a conflict of interests on those occasions where a defective deliverable might need to be held back from delivery by QA. Personnel in QA and Testing ideally need to be self-starters who can initiate and follow through on their tasks. They need excellent analytical skills and be willing to "play the detective" on occasion since they will often come across problems and issues that may not be immediately obvious. Being proactive, and able to head off the big problems before they happen is an important aspect of the QA function. Excellent written and verbal communication skills are also required. Whether performing QA tasks or testing, the individual needs to communicate findings to other developers as well as to management. This skill cannot be emphasized enough. The ability for the QA/Tester to communicate findings and make recommendations to peers and to management through written reports, discussions, and presentations will determine much of the his or her success in that role, and ultimately much of the success of the QA and Testing organization. The QA specialist also needs to be personally convinced of the importance of quality assurance and testing. Without the enthusiasm to support their activities, it is extremely doubtful that the QA and Testing function will be successful. Doubt and negativity will only reinforce a perception that the QA and testing activities are not as value-added as programming activities and are a drain on schedule and budget.

It's important that the level of reporting for the QA and Testing organization be high-level since this gives it the independence and authority to deal with other functions on an equal basis. Our QA and Testing organization has reported over the years to the Chief of Operation (COO) and the Chief Executive Officer (CEO). Currently, our head of the QA and Testing organization is the Director of Quality Assurance who reports directly to "Executive Management" which includes the Chief Financial Officer (CFO), the COO, and the President of the company.

Building a QA and Test Organization from Scratch

Benoit

It was important to define the perceived purpose and goals of the QA and Testing function. In other words, what did we want QA to accomplish? Our goal was that QA would help to ensure that the software development process was being complied with through ongoing QA audits, reviews, and management notification. Having a specialized function to provide management with visibility of project processes meant that the development team members were freed up to concentrate on what they did best. In addition, project management personnel had a dedicated extra set of eyes and ears ready to alert them to problems and deficiencies early on in the project life cycle - or at a level where they wouldn't normally be noticed until the impact was much greater. Our QA and Testing process would parallel the software development life cycle activities and would focus both on prevention as well as early detection of defects.

Step 5 - Select the Pilot Project

The next step is to select a suitable pilot project into which to integrate the QA and Testing activities. It is necessary to have a tangible project structure with which to work. Often, the arrival of an important project is a driving factor in establishing a QA/Testing function, as is a desire or need to achieve SEI CMM compliance or ISO 9000 certification. Our pilot project, or case-study, was a major Oracle full-life cycle development project that would use the Designer/Developer tool set, and which was planned to last at least three years - plenty of time in which to establish a QA/Testing function from the ground up. The project was to design and develop a maritime logistics support application in a client-server Windows NT and HP-UX environment, and was to be developed using an Oracle RDBMS and Oracle *CASE 5.1 - which was later migrated to Designer /Developer. Achieving SEI CMM Level 2 was a contractual requirement of the project. Selecting a large project over a small project as the case study has the advantage of scope and time. It's important that you give yourself sufficient time during which to really get to grips with the QA and Test process, and to try out different techniques. We found it extremely valuable to work on such a project since we were able to cover the entire development life cycle.

Step 6 - Integrate the QA function into the Life Cycle Phases

Using our documented software engineering process that required specific inputs and outputs (entry and exit criteria) for each life cycle phase, QA was able to develop a checklist of deliverables to review prior to the next phase being started. Through such activities as consistent inspections of processes and products, implementation of project standards and guidelines, and frequent checkpoints in the life cycle such as reviews and prototype meetings, it was expected that a high level of Customer confidence would be maintained. We felt that it was important to define specific QA and Testing activities that would take place at each stage of the life cycle. There was no question of the QA and Testing function being brought in only at the end of the development phase. QA and Testing activities were to be performed as integral parts of the full life cycle, from day one of the project.

Strategy Phase

Develop the Plans and Procedures

Planning activities for QA and Testing were undertaken during this phase. The first assignment for the QA specialist was to write the project's QA Plan and Procedures. This had to be integrated with the

Building a QA and Test Organization from Scratch

Benoit

Project Plan, comply with WR Systems' software development methodology, and specifically address the development platform and tools to be used. The section on SQA Procedures, Tools, and Records was by far the largest section of the plan, with 49 procedures addressing more than 70 project deliverables. Risk areas were identified and a schedule of QA audits, reviews, and testing was created based on the project schedule. It was important to identify how testing would be accomplished, and to describe specific QA procedures that would be applicable to the many stages and activities of the project. The project's draft Test Plan and Procedures was also written at this stage. The test schedule, test methodology, and planned tests were described at a high level, based on requirements. Testing techniques would include software inspections, code walkthroughs, peer reviews, audits, and dynamic testing. Test phases would include unit testing, integration testing, system testing, and acceptance testing.

Maintain Quality Records

Documenting the results of QA audits and reviews, as well as testing, is extremely important in establishing an audit trail of QA activity, compliance with SEI CMM, and follow-up on corrective action. Every QA and testing activity was documented on a pre-printed form specific to the particular procedure. All defects from the product were documented with the required corrective action. The form was completed on-line as a Word document and then printed for storage in the Project Notebook, and for distribution to Project and Corporate Management, as required. The QA records were maintained both as hard copies and electronic files, and were available for review by the Customer at any time. They became a valuable tool in maintaining an audit trail of QA activity, in particular deliverable reviews, and were referenced many times in order to settle any confusion over what was reviewed, by whom, and when.

Keeping Metrics on QA Activities

The QA records satisfy the requirement to document QA activity. However, we decided to take this a step further and find some method of easily producing reports and queries on the metrics we were keeping; for example, the results of audits and reviews per subsystem or deliverable, defect types, density of defects, frequency of certain types of QA activities, etc. Using Microsoft Access, QA decided to develop a simple database application with data entry forms and reports. The application was called the *Quality Assurance Tracking System (QATS)*. Data from QA reviews and audits were entered on a regular basis, providing a good basis for metrics reporting throughout the project, such as the type and number of problems or defects encountered in QA audits and reviews. This information was particularly effective in identifying areas for improvement or corrective action. For example, during the early phases of the project, we found that the majority of defects overall appeared to be in the areas of documentation and non-conformance to standards. Monthly reports also provided listings of all the QA activities for the project - something that's always useful when trying to get a picture of how QA is providing value-added project support.

Reviewing the Requirements

One of the most critical elements of a successful project is the quality of the requirements. Time spent in reviewing the requirements at the beginning of the project will pay dividends as the project progresses. During the strategy phase of the project, the requirements were carefully reviewed by the QA specialist, who defined subgroups and unique requirement IDs which identified the subsystems and functional

Building a QA and Test Organization from Scratch

Benoit

subgroups to which each requirement belonged. The emphasis for QA was on reviewing and organizing the functional requirements (170 of them) into the functional categories that would be associated with the various application subsystems. Once the requirements were organized, QA reviewed them for clarity, redundancy, completeness, and testability. This was the first step in the creation of a Requirements Traceability Matrix (RTM) which would become one of the project's primary QA tools for helping project management to keep the development process on track and of ultimately providing assurance of test coverage. Specific features were associated with the requirements:

- ?? Each requirement had to be specific enough to be testable and had to specify *what* needed to be done, not *how* to do it.
- ?? Each requirement had to be uniquely identified so that it could later be traced to functions, modules, code, and tests.
- ?? QA would ultimately verify that each requirement was implemented in the finished application, and that every feature of the finished application corresponded to a requirement.
- ?? Defects in an application would be categorized as missing requirements, wrong functionality, and extra features (i.e., not corresponding to a requirement). A requirement not implemented is a failure to provide the agreed-upon system.
- ?? Functionality with no corresponding requirement is an error in scope ("creeping featurism" or "gold-plating") which may result in an increase of schedule and budget.

At this stage, we were able to start to design basic high-level functional tests. The requirement IDs were used in generating reports and in tracking test case coverage.

Develop the Requirements Traceability Matrix (RTM)

The RTM is a vital part of QA's toolkit. It allows QA to monitor the progress of requirement coverage throughout the project and ensure compliance to Customer requirements of the software and its deliverables. We decided that the development and maintenance of the RTM, a key requirement of the company's software development process, would be formalized, and that the QA specialist would take overall responsibility for its development. This was extremely useful in keeping the QA specialist "in the loop" regarding requirements – something that is important in ensuring that you end up with well-defined, testable requirements. Finding a repository for the requirements was the first step. It was necessary to maintain the customer requirements electronically, both to more easily maintain the inevitable changes over the project life cycle, and to be able to print out reports and trace a history of changes to requirements. We chose to implement the RTM through the Oracle database, but it can also be implemented, as we have done on subsequent projects, in other ways, such as with an Access database, Excel spreadsheet, or Word document. The important thing to remember is that a formal and consistent method of tracing the development products to the requirements is essential. The RTM was pivotal to the testing process at every stage of the life cycle from Analysis through Build and Test. The QA specialist became the custodian of the Functional Requirements List (FRL). All requests to change the list – initiated either by users or by the developers - had to be approved through a formal Configuration Control Board (CCB). The requirements change management procedure was developed by QA and the Technical Manager to ensure that we had in place the essential control, while retaining the flexibility for the customer to make modifications, as required. The request for a change to a requirement was

Building a QA and Test Organization from Scratch

Benoit

submitted via a pre-printed form developed by QA. The form was emailed to QA who then reviewed it for completeness and clarity before submitting it to the Configuration Manager (CM) who called the CCB meeting. Approved changes to requirements were subsequently entered into the RTM database by QA who reprinted the Functional Requirements List (FRL) with the most current requirements. This was distributed to team members and to the Customer. The tight control over the requirements proved invaluable over the three-year project cycle. A history of changes could be easily printed as a report from the RTM database at any time, as well as an up-to-date FRL. This activity was perceived as a definite "value-added" activity by the QA/Testing function.

Establish the Deliverable Review Process

Independent review of a deliverable product is a Quality Control activity that can catch defects that the producer has missed. Every deliverable on this project was to be put through a formal review procedure that included a Technical Review followed by a QA Review. Once it was clearly established that all deliverables would be put through a review process, review route sheets were developed by QA that would accompany each deliverable on its journey from the developer to the technical review, to the QA review, and finally to the customer. The route sheets were checked off and signed by each reviewer once the review was considered complete and the product satisfactory. The Technical Review, generally performed by the Technical Manager, would ensure that the product was technically correct and effective. QA would then review for compliance with standards, procedures, completeness, and overall professionalism. Once all the project members were instructed in the review process and underwent it a couple of times, it became second nature. This was one of the more successful QA procedures and was of course, particularly necessary during the analysis and design phases where the greater portion of the deliverables consisted of design documents and plans.

Create the Project Standards and Procedures

Project standards need to be set up during the early stages of the project. The Technical Manager wrote an extensive and comprehensive manual of project standards which covered everything from data naming and coding standards, to documentation, and Software Development Folder (SDF) standards. QA reviewed and contributed toward the standards, and the project team was trained in the use of the standards document. Looking back, the creation of such extensive and detailed standards was one of the most useful activities that was undertaken to ensure consistent, high quality development. As new people joined the team, QA was able to give them this Project Standards document and train them in the specific standards appropriate to their task.

Analysis Phase - Begin the QA Reviews and Audits

The analysis phase was the start of a period of ongoing auditing and reviewing by QA for verification of compliance with requirements, standards, and procedures. The effort spent in these activities at this early stage was extremely worthwhile, since defects at the early stages of the life cycle are much cheaper to find and fix than those detected during the later phases of development and testing.

The deliverables from this phase included:

Final Requirements Document	Entity Relationship Diagram(s) (ERD)
Function Hierarchy	Data Dictionary

Building a QA and Test Organization from Scratch

Benoit

Traceability Matrix (Requirements-Functions)	Create Read Update Delete (CRUD) Matrix
--	---

Design Phase

The design phase proved to be one of the busiest phase for QA on this project. There were many deliverables developed during this phase that needed to be reviewed and tested.

Physical Data Base Design	Module Network (Menu) Hierarchy
Physical CRUD Matrix	Module Specifications
Data Dictionary	Updated RTM (Requirements-Functions-Modules)
Prototype Model(s)	

During this phase, QA actually performed detailed reviewing of all deliverables and report output to ensure that standards had been followed. Project Data Naming standards were strict and QA developed several scripts to help detect non-conforming names. The prototypes were tested by the developers as well as being independently tested on an ad-hoc basis by QA.

Design Reviews

Design Reviews with the Customer are Quality checkpoints in the SDLC. It's an opportunity to elicit formal Customer feedback on the application design. The QA specialist attended both the Preliminary Design Review (PDR) and the Critical Design Review (CDR) and took detailed notes as a form of verification that all action items were documented and subsequently resolved. Action Items from these meetings were entered into the Problem Report/Action Item Tracking Database - a small Oracle database that was developed by the CM manager as a repository for problem reports and action items. The QA specialist periodically reviewed the database to ensure that corrective action was taking place and that all action items were addressed. Supporting documentation and deliverables for the design reviews were put through technical and QA reviews prior to being delivered to the Customer in time for the meetings.

Build and Test Phase

Peer Reviews and Code Walkthroughs

The Build phase is an appropriate stage to introduce Peer Reviews and Code Walkthroughs as team activities. Both are extremely effective as QA processes and help to remind the developers that Quality is their responsibility. As part of our software development process, these procedures were already documented and therefore relatively easy to implement. The QA specialist attended these activities on an ad-hoc basis, essentially to verify that they were being performed correctly and that they were documented. Participants in these activities were trained to perform their assigned roles as reviewers, facilitators, recorders, and producers. Items for review were distributed to the participants in sufficient time prior to the activity to allow at least an hour of review time. The peer review or code walkthrough would generally last about an hour; any longer proved ineffective.

Develop Prototypes

As the development process progresses, it is important to keep the Customer informed on the progress of the application. We found that prototyping specific functional areas at regular intervals served to both assure the Customer and to verify that we were still "in synch" with Customer requirements. We held bi-weekly "Prototype Meetings" with the Customer to demonstrate the completed modules. These meetings

Building a QA and Test Organization from Scratch

Benoit

were extremely successful in maintaining a high level of visibility between the Customer and the Development team and helped to iron out any remaining problems concerning functionality and design. The initial stage of the Build phase actually overlapped with the final phase of Design since this was considered an effective method of showing the Customer what the system would look like in real life. Each Prototype was packaged with the software modules, supporting User Documentation, which was written in parallel with the software development, various Oracle-generated reports, and associated Requirements' Traceability reports. QA designed a route sheet that itemized all the required objects that belonged to a specific prototype. Every object was put through the Review Process with final sign-off by the QA specialist. Action Items that came out of the Prototype meetings were entered into the project's Problem Reporting/Action Item database. QA would periodically verify that action items from specific prototypes had been resolved and closed out. If not, a Management Notification Report, highlighting the problem, would be written up and passed to the Project Manager and Technical Manager. At this stage, the testing of prototypes was performed by the developers, as unit tests. The prototypes were also demonstrated to the Technical Manager as a technical review. The QA specialist was concentrating on reviewing the supporting documentation products.

Unit Testing

Unit Testing is traditionally the domain of the Developer. The unit testing was at first, very informal. In the ideal world, where time and money isn't an issue, this would not have been a problem. As a result of our Prototype demos, however, it became apparent that informal unit testing was not being performed in the most consistent or appropriate manner. Tight schedules often squeezed out the unit test stages and developers weren't trained in testing and test writing. They would often test their modules to prove that they worked and spent little or no time on negative test cases that would find the bugs. With a large number of modules to be developed and tested, and with integration testing coming up, it became imperative to get tighter control over the test process. We addressed this by having the QA specialist develop and teach a basic Test Writing course for the developers. The emphasis of the course was on building effective test cases that would be designed to find bugs. The test quality improved noticeably after this training. Having identified unit testing and later, bug fix testing as areas of weakness, the QA specialist initiated a more formal method of unit testing using documented unit tests, peer reviews, and with the CM manager and QA specialist more tightly integrated with the process. The QA specialist now performed independent testing on an ad-hoc basis to ensure that the tests were fully exercising the module's functionality and that test results were correct. A test form was developed by QA to serve double duty as the test script, the test report, and signoff form. The new-style unit test required signoff by both a peer reviewer and QA. Only then, could the module be considered to have been successfully unit tested and moved to the appropriate location by the CM manager. The completed test forms were kept in the Software Development Folders (SDFs) by the developers. QA kept copies of the test forms and used them to develop statistics such as the number of bugs found during testing, and the percentage of tests that were failed by QA or by the Peer Reviews. The initial disadvantage that we found was that the developers tended to rely on QA to catch all their errors. However, following some discussions in project meetings and a period of "spot checks" by QA where the developer and/or the peer reviewer was required to demonstrate to QA how they had tested the module prior to their signing off on it, the situation was resolved to everyone's satisfaction.

Building a QA and Test Organization from Scratch

Benoit

Integration Testing

Decision to Automate

Compuware's test management tool *QA/Director* was selected as a tool that would allow tests to be maintained in a test repository - a definite advantage for regression testing - and which would track bugs/problem reports, provide test execution reports, and show test progress - all in an integrated package. Reusable test cases was considered to be a definite plus. QA configured and installed the test tool and trained developers on how to write unit tests and enter them into the test repository. We found that developers were far more likely to develop good tests when using this tool. The tests were reviewed by the team leaders, technical manager and QA. Test scripts were printed out to supplement test plans and other test documentation; Test cycles were set up that included entire test suites within subsystems of the application under test; Bugs were entered into the Bug Tracking repository and assigned to specific developers for action. QA was able to go into the tool, run tests, review test results, and monitor bug fix progress. Compuware's *QA/Director* enabled us to provide structure to our test process although testing remained essentially a manual process since the testers still ran the tests step-by-step, rather than by running automated scripts.

Developing the Tests

Test development was accomplished in two phases.

1.	Writing the High-level test descriptions	User business scenarios, collected from the Customer, were used as a starting point for determining interaction and integration between modules. QA worked the scenarios into basic test descriptions that encompassed the integration of the associated modules. With over 700 modules and a complex functionality among six subsystems, this was possibly one of the more difficult stages of testing and required a great deal of interaction between functional users, developer team leaders, and QA. The need to integrate the software with several external systems also needed to be addressed.
2.	Writing the test scripts	Using <i>QA/Director</i> , the developers then wrote the test cases and test scripts from the test descriptions. We monitored the test writing progress and coverage through the various graphing and reporting features of <i>QA/Director</i> . QA continued to monitor test coverage through the RTM.

Managing the integration test effort

A test team was formed consisting of selected developers from each subsystem and a senior developer became the Test Manager. In order to maintain independence and be able to monitor the test process, the QA specialist did not participate in actual test execution. From time to time, QA would perform random independent testing or would monitor actual test execution by the testers. This proved to be a successful method, particularly at times when a pressing schedule required speedy execution of tests. Test meetings were conducted daily to discuss test schedule, progress, and problems. Using the Test Plan's hierarchy of test dependencies, test cycles were scheduled that included specific modules from the various subsystems. QA was the final sign-off on the successful completion of test cycles and all defects found during testing were logged into the bug tracking database in *QA/Director* for resolution by the developers.

System Testing

Building a QA and Test Organization from Scratch

Benoit

The project used a mixture of “requirements-based testing” and business-scenarios test cases, which meant that we needed to verify that all the requirements previously specified were actually supported. Tests to determine compliance with the requirements were therefore written with the requirements in mind. All test scripts and test scenarios had to specify the requirement(s) being tested. If any additional test scripts were written, or existing scripts updated, requirement references had to be included. In order to facilitate the development of the tests, QA designed a Test Scenario form which enabled the test writers to include pertinent data for their tests. We used Compuware’s *LoadRunner* to assist with Performance and Load Testing. This allowed us to perform volume and peak load testing before the production hardware was available.

Deployment Phase

In order to facilitate deployment and cause as little disruption as possible to the users, we undertook a phased deployment, releasing individual subsystems several months apart. Our main challenge was to maintain two environments running in parallel – the development/test environment and the production environment. Strict CM control of source code in both environments was vital at this stage, as was ensuring that any modified code was kept in synch. QA review and test procedures continued as normal in the development environment. In the production environment, however, a faster turnaround was required. To facilitate this, problem reports and change requests from the users were called in or faxed, and the QA and CM functions worked closely together to expedite the test and delivery of modified code. User training took place prior to implementation, and QA worked closely with the trainers to develop and test all training materials.

Maintenance Phase

Once the system was deployed, the process of ongoing "tweaking" continued as the users became more familiar with the new application. The QA, Testing and CM procedures that were put in place during development were kept in use once the system was deployed. The source code remained under configuration management and any change requests from the user community were processed according to the same CCB review process as during development.

Step 7 – Use Lessons Learned

Acceptance of QA and Testing - both the function and the person

It’s vital that the QA and Testing function is perceived as both collaborative and “value-added”. Someone who is confrontational with the developers, overly and personally critical, and who “points the finger” and apportions blame, is going to damage the reputation of the QA and Testing function - possibly for good. Perceptions, although often incorrect, are unfortunately, often impossible to eradicate. The QA specialist was always made to feel part of the team and was given responsibility for such tasks as the RTM and Requirements Change Process - activities that were seen as “concrete” and “added value” rather than just adding to a perception of “QA as police” checking up on everyone. Avoiding the “them and us” syndrome is critical in making QA a success. It is equally important that there be a structure to the QA/Testing activities rather than a sense of free-floating around the project looking for problems. This is particularly important when the QA person is billable, and accountability is an issue. Well-designed QA and Test Plans, detailed and specific QA/Testing procedures, and well thought out QA and

Building a QA and Test Organization from Scratch Benoit

Test Schedules are tools that are invaluable in providing structure, guidance, and credibility to the QA function. Specific tasks related to the QA function need to be clearly defined for each life cycle phase.

What Worked?	What Didn't Work?
<ul style="list-style-type: none"> ?? Deliverable Inspections by QA/Tester - QA established as final sign-off on deliverables ?? QA/Tester involved in requirements review and management from the very beginning ?? QA/Tester involvement in Requirements, Design, and Code Reviews ?? QA review forms and QA records; Centers of Excellence (COEs), Code Walkthroughs, Peer Reviews ?? Formalized and documented Unit Testing ?? Using a database to facilitate the creation and maintenance of the RTM and requirements' coverage reporting. ?? Using a COTS test management and bug tracking tool during integration test ?? Entering all problem report, defects, and action items into a database throughout the project 	<ul style="list-style-type: none"> ?? Informal unit testing ?? Anything involving excessive documentation or paperwork by the developers, e.g., complex test forms ?? Anything that caused lengthy turnaround time for deliverables as a result of QA/Testing activities ?? Expecting developers to read and digest lengthy standards documents without guidance ?? Assuming that developers would enter all the required RTM information ?? Assuming that developers would be able to write effective test cases without training

Step 8 - Implementing QA and Testing on all projects – "Clone the Process"

Once QA and Testing had been successfully integrated into a large and lengthy project, it is much easier to “clone” the process on other software projects. Much of the core work has already been done. You have your “Lessons Learned”. You know which QA and testing activities were successful and which weren’t, and where training is required. You have a stash of tried and tested artifacts - audit and reviews forms, standards checklists, procedures, test documentation, plans and guidelines, training and presentation materials, records, repositories, and tools. Some of the more successful documents on our project turned out to be test forms, peer review forms, and QA review and audit forms. Much more than this, you have documented results and have confidence in your processes. Not everything works. Not everyone will “come on board”. But little by little, you will see the results of your efforts as people in the organization take for granted the QA and Testing function as a part of their project. And their creativity is not affected one bit!

Step 9 - Expanding the QA and Testing Organization - Build the team

Following the success of the project, we were able to expand the QA and Testing organization in order to be able to support other projects. The original QA specialist, who was now experienced as well as professionally certified in Quality Assurance and testing, became the head of the QA and Testing organization. When it became time to hire more people for the QA and Testing organization, we used a mix of part-time and full-time people. One solution often cited is to take developers and give them a

Building a QA and Test Organization from Scratch

Benoit

temporary stint in QA and Testing. This, apparently, is an ideal method of ensuring that QA personnel have a technical background and also of ensuring that eventually all your development staff understand and carry the torch for all those QA ideals. However, this method rarely works in real life. It's a rare developer who will give up a chance to be "creative" in order to endure a period in the dreaded QA and Testing organization! In our QA and Testing organization, we found it easier to take extremely smart people who displayed excellent analytical skills, paid attention to detail, were persistent, and able to concentrate for long periods of time on the kinds of things that would drive lesser mortals crazy. We found that many people were able to be trained in good testing techniques whether or not they had a substantial background in software development since the majority of our independent testing tended to be functional or behavioral testing. In addition, it helped if they were fast learners, self-starters, and had - or developed - thick skins! Some full-timers we hired had some experience of QA or testing already. We had to train them in Quality Assurance activities as well as in Testing. Each QA/Tester is assigned to multiple projects. Project assignments take into consideration the experience and skill level of the individual in order to best support the specific project. Some part-timers were already employees of our company and were talented, smart people who welcomed the opportunity to learn something new and have the opportunity to work in the software engineering arena. Some have enjoyed QA work so much that they eventually became full-time QA specialists or testers. Others moved into software development (although this, in my opinion, is a reversal of what should be happening!) Recently, as the reputation of, and respect for, the QA and Testing organization has grown, there have actually been developers who have requested to join the QA and Testing organization!

Step 10 – Training and Certification

It's important that the QA and Testing organization have professional credibility. Everyone in the QA and Testing organization receives training - a 10-module internal training course in Quality Assurance and Testing that provides a solid foundation for performing their tasks. After about six months of on-the-job and internal training, we encourage the QA/testers to become professionally certified. The first certifications addressed are usually the Quality Assurance Institute's (QAI) Certified Software Test Engineer (CSTE) and Certified Quality Analyst (CQA) certifications since the requirement for years in the field is less than the American Society for Quality (ASQ). Once the QA/Testers have those certifications under their belts, they are encouraged to take the ASQ's Certified Software Quality Engineer (CSQE) examination. This has an excellent "all-rounder" body of knowledge for the Software Engineering industry.

As a small company, it's not always possible to send everyone on some of the more expensive external courses. However, we have usually been able to send each QA/Tester to at least one external course, seminar or workshop each year. We use the "train-back" method, whereby the employee who has attended the course, then trains others internally. In this way, we are able to maximize the benefits of training by giving the original attendee the opportunity to both consolidate his/her training while honing presentation skills, and allows others to benefit from the knowledge.

Building a QA and Test Organization from Scratch

Benoit

Keep up the Momentum

Once the QA and Test organization has been established, and supports all projects, it's still not safe to rest on your laurels. There are constant new challenges in the world of software development and testing. More and more projects are internet-based, development web applications in a Rapid Application Development (RAD) environment. The industry is always changing and the QA/Testers need to stay abreast of the new technologies and methodologies. Maintaining the morale and enthusiasm of the organization, and providing opportunities for training in industry best-practices is extremely important for both the employee and the company.

What out for:

- ?? Cost concerns over whether QA and Testing personnel should be on overhead
- ?? Boredom – try to vary the types and lengths of projects to which you assign QA/Testers
- ?? A tendency for the developers to assume that the QA/Testing personnel are responsible for Quality
- ?? Tight schedules squeezing the test process

Conclusion

The project in our case study was successfully deployed and became an example of a full life cycle software development project with integrated quality assurance and testing. It became our best example of how well a project could do when compliant with a good QA and test process. By the end of the case study project, we had been externally assessed as being SEI CMM - Level 3 compliant. Subsequent projects have gone on to utilize the same QA processes and procedures, and the QA and Testing organization has grown to support all software projects at several geographical locations.