



Assuring Software Quality or Is it good enough yet?

Bruce Paynter, CSQA

OVERVIEW

Software teams develop software
Software development teams develop software, mostly highly customized software, to meet specific business needs. These teams also enhance the software to meet major changes in the business needs. Software maintenance teams maintain the software, fixing problems and sometimes making small refinements to meet minor changes in the business needs. Both teams must produce software with a degree of quality which the business views as acceptable for use, within a certain timeframe and at a certain cost. The ability to do this distinguishes successful teams from others.

KNOWING WHEN THE SOFTWARE IS READY FOR USE

Successful teams know when the software is ready for use. These teams will not release software until they have assured themselves of its quality. That it will work properly, consistently.

Quality Assurance is a management process which uses quality controls to determine when a software product is ready for use. Both teams can use the controls to provide assurances about the software's quality. These controls should be part of the normal work processes followed by the teams. Successful teams have established the controls which they need to manage the software quality.

Putting the Assurance back into quality assurance
This article is about putting the assurance back into quality assurance. It helps define what is needed to know when software is it good enough to go into production. It also describes what controls can be used to determine these assurances and presents ways in which this can be done at a low cost to the project.

WHAT IS GOOD ENOUGH

Measurable Targets

The Quality Assurance process must be able to answer the question: "Is the software ready for use?" In order to answer this, the process must get certain information from the established project controls. Information such as what parts of the software have been tested, how many defects were found, the distribution of these defects within the software and what impact do the defects have on the software. If the software is not ready for use, the process should also be able to provide how long until the software is ready.

Test Coverage

Test coverage is a measure of the software which is tested by a suite of tests. Coverage can be examined by business functional requirements or by the physical design of the software code modules. Different types of tests address the type of coverage provided. For example, User Acceptance tests address functional coverage. Unit and System Integration tests address software code coverage. Knowing the coverage leads to a better understanding of what has or has not been tested.

Defect Density

The results from the testing process are either a pass or a defect, maybe multiple defects. Organizing these defects by software code module or business function provides a view of the concentrations of defects within the software. This can be used to identify error prone code modules or a poorly understood business function. A complete lack of defects may be an indicator of effective tests.

Defect Severity

Categorizing defects by severity to the business provides the team with the ability to understand the

impact the defects will have on the business. Most businesses can still use software if it contains several cosmetic defects, but it only takes one critical defect in which using the software could have catastrophic affects on the business. Knowing the severity of the outstanding defects helps the team determine the readiness of the software.

Effort to Correct

It takes time and effort to correct defects. By indicating the amount of effort and time to correct each defect, project management can understand the enormity of the effort and can organize the team to correct the most critical defects within the project schedule.

WHERE DOES THIS INFORMATION COME FROM

This information can be found in several places within a project. Here are the most common areas which may have the needed information.

Reviews

Most teams conduct reviews of documents, such as user requirements, system architecture and test plans, to name a few. Reviews are also done on work products such as data flow diagrams, code modules and test cases. These reviews vary in formality from the informal “please provide comments” document reviews to very formal design walkthroughs. Reviews are meant to identify problems areas to be addressed by the document or work product author before proceeding. Sometimes, teams keep a record of the problems identified in the form of a comment list or an action list for the author to address.

Inspections

An inspection is a formal process with designated roles, such as inspectors, moderator and recorder, assigned to each participant. Inspectors and authors use a checklist to inspect a document or work product to identify the defects. The recorder records all defects. The author can then use the defect list to improve the document or work product.

Dynamic tests

Dynamic tests execute the software to see that it performs as expected. Dynamic tests are normally grouped into Unit tests, Integration tests, System level tests and User Acceptance test based on who is involved and the status of the software within the

development life cycle. For example, Unit tests are conducted by the program code author and are focused on one unit of code only. After passing the unit tests, the code is ready for integration with other units of code. Integration tests examine the assembled units of code to see how well they work together. These tests are normally conducted by designated testers, who send the test results back to the code authors for corrective action. After passing the integration tests, system level tests are conducted and then finally the User Acceptance is conducted. Test results are normally recorded for all tests except unit testing.

HOW TO GET THERE

The following is a list of the major activities required to support QA within a project:

1. Determine the target measurements which will be used to monitor the project progress and the product readiness. Use snapshot measurement at the critical project decision control points to understand the current situation and future scenarios. Use monitoring measurements to determine the progress towards the next decision point.
2. Establish the necessary quality assessment and controls early in the project life.
3. Integrate these controls into the existing work practices followed by the team.
4. Provide training on the modified work practices to all affected team members. As a minimum, this should cover inspections, formal reviews, dynamic test processes, change management and estimation.

LOWERING THE COST OF QUALITY ASSURANCE

One of the major concerns organizations have about QA is its cost. If QA is established as an add-on activity to the project teams workload, these costs can be expensive. Adding any new processes to a workload, costs resources. Separate QA processes are no exception. Modifying the existing work practices followed by the team to include the controls reduces this cost. In return for these changes, the team has the information needed to determine the software’s quality before moving it into production.

Early detection is key

A lot of teams use the test results from the dynamic tests to determine when the software is ready. Unfortunately, this information is available very late in the development process. Getting this information when the project is about 90% complete leaves very little time and resources to react to a low quality report. A project in this situation will experience schedule and cost overruns to correct the defects. At this stage in the project, the only alternative is to correct defects after putting the software into production.

Identifying these defects earlier in the project would provide the team with the opportunity to correct the defect at much lower cost and be able to adjust the schedule. Obviously, the dynamic tests cannot be run before the software is constructed. By implementing a combination of inspections and formal reviews permits the team to assess the quality of the requirements, logical design, technical architecture and physical design before construction even starts.

Define a Strategy

Another way to reduce costs is use a strategy to focus the team's resources and vary the test coverage to get the most 'bang for the buck'. The following is a list of some of the most common strategies:

1. Most critical business functions first

2. Highest frequency of usage first
3. Widest component coverage
4. Most error prone module first
5. Weakest Technical Component

SUMMARY

Successful teams have the ability to determine the level of quality of the product before delivering it to the client. Testing helps determine the level of quality and all programming staff conduct some sort of testing of the software they build. This includes code unit testing, integration testing, functional, system, stress and operational tests, to name a few. The results from these testing efforts help the teams determine the quality of the software. Inspections and formal reviews of the requirements, logical design, technical architecture and the physical design before the code construction begins gives the team the ability to remove the known defects before the repair costs escalate.

Modifying the teams existing work processes to provide the needed controls and measurement information lowers the cost of assuring the software's quality. Defining what is meant by 'good enough' for the software to be acceptable for use by the business enables the team to focus on developing the software rather than the controls. This is what separates successful teams from others.

About the author

Bruce Paynter, Senior Management Consultant, Q/P Management Group, wrote this article. Mr. Paynter specializes in software measurement, quality assurance, development methodologies, and process improvement. He is an experienced international consultant and instructor in the areas of function point analysis, quality inspections, quality assurance planning and strategies.