http://www.sdtimes.com/GUEST_VIEW_QUALITY_ASSURANCE_MEETS_QUALITY_CONTROL/By_Robin_F__Goldsmith/About_QA_and_TESTING/34520

Guest View: Quality Assurance meets Quality Control

By Robin F. Goldsmith

August 1, 2010 — (Page 1 of 3)
Students and clients often ask what the difference is between Software Quality Assurance, Software Quality Control and Software Testing. The classic distinction is that Quality Assurance (QA) addresses the processes that produce software, whereas Quality Control (QC) deals with the products of those processes. Testing is the primary method of QC for software.

That said, these definitions routinely are interpreted in a variety of ways that often are inconsistent at best. For example, most "QA" software groups actually only do testing.

For those relatively few QA groups that are not just doing testing, probably the most common distinction organizations make is that QA reviews requirements and/or designs whereas QC/Testing executes the developed software programs.

It should be evident that this definition of QA as reviewing requirements/designs does not actually address the software process. Review is a form of static testing, albeit typically of intermediate development products earlier in the software process. Executing code is dynamic testing of the end product of software development. QA vs. QC distinctions based on what is tested blur further when it’s the code itself that is reviewed.

Just to add a bit more fuel to the fire and mix metaphors, many people muddy the waters more with the term verification and validation (V&V). For many, verification (that the product was created right) refers to static reviews, and validation (that the right product was created) refers to dynamic code execution tests.

In fact, most dynamic test execution, regardless if it's called testing or validation, is confirming only that the developed software conforms to its design, not that the design was right, and especially not that the requirements were right. This shortcoming is made even more likely by the common mistake of referring to the design as the requirements, which usually signifies that the REAL requirements have not been identified adequately.

I recognize V&V is a well-established practice, especially for Department of Defense development, and often is not founded on static vs. dynamic testing distinctions. Many of those performing such V&V very much do endeavor to validate that the right product is being created. Nonetheless, I still find the term "verification and validation" too often is used in ways that don't add to my understanding, and I personally avoid it.

August 1, 2010 — (Page 2 of 3)

Traffic cop view

Even those who view QA as more than reviewing requirements and/or designs often still consider QA largely as sort of a traffic cop concerned with catching discrepancies after they've occurred. For instance, IEEE Std. 12207-2008 on Software Life Cycle Processes says, "The purpose of the Software Quality Assurance Process is to provide assurance that work products and processes comply with predefined provisions and plans."

The standard goes on to say that the QA process should be coordinated with the related software verification, software validation, software review and software audit processes; problems or non-conformances with [formal or informal] contract requirements should be documented and serve as input to the problem resolution process; and records of these activities, problems and problem resolutions should be maintained.

IEEE Std. 12207-2008 also makes QA's frame of reference the project "contract" and its related plans, . QA assures that products and processes comply with that project contact. As used here, the contract represents an agreement between an internal or external supplier and the customer/user/acquirer for whom the software is being created, operated and/or maintained. The standard says QA assures the contract is documented, plans are consistent with the contract and adhered to, and products fully satisfy contractual requirements and are acceptable to the acquirer.

Although IEEE standards' picture of QA is broader than merely reviewing requirements and design documents, I find it still too reactive and restricting to QA's real value.

Proactive SQA

SQA can provide far greater payoffs when it becomes a positive, proactive part of the process producing systems and software. In my consulting and seminars, I describe Proactive SQA that establishes an environment that promotes quality and covers the full life cycle of development from conception through retirement. While checking various work products can be part of its role, Proactive SQA emphasizes analyzing and improving the processes that create as well as control quality, including development methods, workmanship, management practices, and error prevention.

Thus, rather than merely confirming compliance to contracts and plans, Proactive SQA first assists in ensuring those contracts and plans are accurate, complete, and reasonably likely to produce necessary quality. Moreover, Proactive SQA helps define effective development and quality control processes, and it evaluates them in practice to make sure they deliver and continually improve.

August 1, 2010 — (Page 3 of 3)
Proactive SQA ensures that six key value-adding functions are done well, . That doesn't mean that the SQA group necessarily does them personally, of course, but merely that they're being done somewhere and by someone:

1. Define Quality Assurance Plans identifying what to do to create and ensure quality.
2. Define standards, conventions, guidelines and other practices and techniques—along with their associated learning and guidance.
3. Ensure systematic quality control of processes and products so that it gets done right. Such quality controls include both static and dynamic testing as well as configuration management/control, production release, project management/control, supplier contracting and management, document control, operations and support, maintenance, backup and recovery, and security.
4. Maintain quality records to keep track of it.
5. Analyze and report on quality to learn from it.
6. Direct attention to maintaining and improving quality to encourage it.

Together these help the organization better do the right things correctly, which actually is the cheapest and fastest way to create and maintain necessary quality.

Robin F. Goldsmith is president of Go Pro Management and author of Discovering REAL Business Requirements for Software Success.