QUALITY ASSURANCE (QA) IN SOFTWARE TESTING: QA VIEWS & BEST PRACTICES

Software quality can be hard to determine. That's because each person involved can define quality differently. A range of perspectives, from different stakeholders and entities, may measure software quality in terms of how it fits to their own requirements, expectations, and standards.

Quality Assurance (QA) is a common practice to ensure that the end product of any Software Development Lifecycle (SDLC) conforms to the overall and scope-agreed expectations.

In this article, we will discuss the basics of Quality Assurance (QA) and its role in software testing.

What is quality assurance?

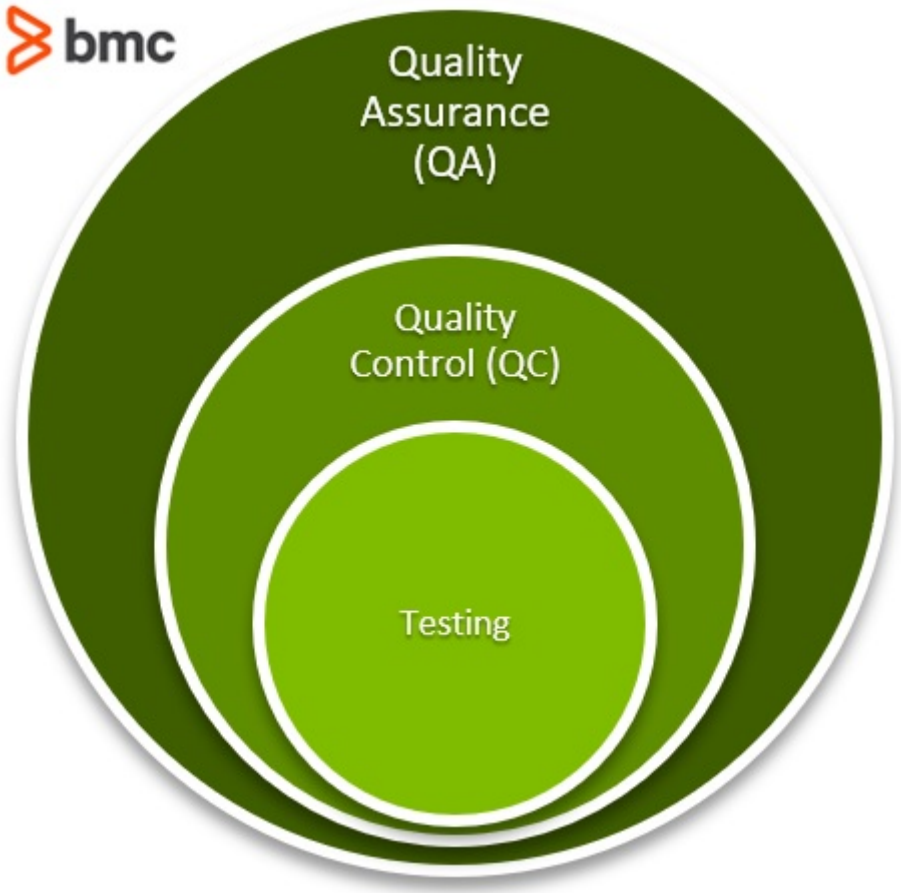
To define quality assurance, let's first start with the definition of quality:

> *Quality refers to the conformance to implicit or explicit requirements, expectations, and standards.*

In order to fulfil these requirements, a quality control mechanism is set up.

Quality Control (QC) is the process through which you achieve, or improve, product quality. The QC process can also include the activities used to eliminate waste processes in the SDLC. QC functions involve a range of testing activities used to detect and resolve technical issues. These are followed together with the development processes over the course of the SDLC.

Quality Assurance (QA) refers to the meta process that ensures continuous and consistent improvement and maintenance of processes that enables a QC job.



(Source)

Quality assurance views

In software testing, Quality Assurance goes beyond quality control functions and testing activities in order to evaluate software quality according to a range of views:

- **Transcendental view.** A notion that we can recognize quality—but cannot measure it.
- **User view.** How the end-user perceives and experiences software quality.
- **Manufacturing view.** If the product is developed right the first time and without flaws during the end-to-end SDLC process.
- **Product view.** How well the internal and inherent characteristics conform to the defined requirements.
- **Value-based view.** How well the product satisfies the requirements of various stakeholders and entities associated with the product.

These different views are the essential part of any QA process. A software build can conform to all necessary requirements and pass the underlying quality control testing process—but all that does not guarantee a positive business impact or improved user experience.

So, we can define QA as this:

The QA process ensures that the wider goal and vision of the business is achieved by delivering software that meets all quality requirements from both technology and business perspectives.

(Explore common software quality metrics.)

QC vs QA

Let's compare the actual testing and QC activities to the QA process, so we can understand scope and mechanism of each:

Quality Control	Quality Assurance
A walkthrough that involves several testing activities	The process of auditing software quality based on different views
Product-oriented activities	Process-oriented activities
QC actions focus on verification and conformance of the product to requirements only	QA actions focus on the process used to create the product
Actions involve inspection, sampling, and testing	Actions involve documentation, audit, management, training, change control and management and investigation
A reactive and corrective process	A proactive and preventive process
The team	The team and relevant stakeholders

For organizations following the modern SDLC methodologies such as DevOps, software testing QA follows the concept of Continuous Improvement, which is the iterative improvement of the processes you use to deliver high quality software.

The improvement is measured against how end-users and the business organization perceive software quality, suggesting areas of improvement. The feedback is then channeled back to the SDLC process where a different or additional set of Quality Control functions may be introduced to address those new requirements.

Quality assurance best practices

These industry-proven best practices can help improve your QA capabilities, the associated SDLC process, and the overall quality of software products in DevOps environments:

- **Automate tests.** Automating tests that are repetitive and require minimal manual intervention increases your speed, agility, and productivity.
- **Shift left.** Ensure that software defects are identified fast and early—shifting left—during the SDLC life cycle.
- **Gather feedback.** End-user and stakeholder feedback should drive continuous improvements of your QA strategy.
- **Test with purpose.** Ensure that testing resources are used in meaningful ways and follow the code deployment structure of the Software Development Lifecycle.
- **Embed security into software quality from the ground-up.** Introduce security-related test

cases as part of the shift left testing strategy.
- **Automate carefully.** Automating a waste process will only create more waste processes.
- **Incorporate human skills.** Automation and continuous improvement don't always work well together to meet DevOps QA goals. For instance, mobile user interface and user experience tests are challenging to automate as they must account for a range of uncontrolled factors, such as human and market perception of the mobile app user experience. As such, human intervention may be necessary as part of an effective QA strategy in DevOps compromising the continuous testing practice inspired by automation.
- **Keep the customer first.** Finally, a DevOps QA must keep the customer view of software quality at the forefront of all QA activities.

Related reading

- BMC DevOps Blog
- How to Write Test Cases for Software
- SRS: Software Requirement Specifications Basics
- Performance Testing, Load Testing & Stress Testing Explained
- What's Testing as a Service? TaaS Explained
- What is Test Data Management and Why Do We Need It?