# Learning Quality Assurance On the Job

Trying to break into the field of software quality assurance can initially feel rather daunting. After just a few exploratory searches online, it's all too easy to feel bogged down by the dozens of industry standards, the slew of complex yet often outdated tips and best practices, and the hundreds of vague acronyms to digest within the software quality assurance world.

To add insult to injury, as a newcomer to the industry, it can be incredibly challenging to land a software quality assurance job without any formal experience already in your back pocket. Many established companies aren't interested in individuals who lack experience with testing or quality assurance of some kind. This paradigm creates the all-too-common catch-22: No job unless you have training and experience, yet no experience can be had without a job.

Thankfully, not all hope is lost! In this article, we'll dig a bit deeper into what software quality assurance is, how it is typically performed, and a few opportunistic ways in which software quality assurance experience can be had for newcomers as well as veterans alike.

## What is Software Quality Assurance?

Categorically defining software quality assurance is a monumental task, given that the practice has been around for decades, in addition to the sheer quantity of unique projects where software quality assurance has played a crucial role in creating a beautiful and functional end product.

At its most abstract level, software quality assurance is simply the means by which the various processes, procedures, and standards of a given project are deemed appropriate and are ensured to be correctly implemented. With proper software quality assurance in place for any particular venture, the entire team can rest assured that the final output, whatever that may be, will be of the highest quality that the group is capable of producing.

## Performing Software Quality Assurance

Software quality assurance is ultimately about defining a set of [software quality attributes](https://en.wikipedia.org/wiki/List_of_system_quality_attributes), which are non-functional requirements that can be utilized to measure the performance of the software, and then evaluating those attributes through a series of meaningful guidelines to ensure quality levels meet a necessary standard. To that end, it is important for any software quality assurance department or individual specialist to have a set of attributes that can be used to properly measure and gauge the qualities of the project being tested.

There are dozens of attribution models used throughout the industry, but one well-known example is [FURPS+](https://en.wikipedia.org/wiki/FURPS), a model originally published by [Robert Grady in 1992](https://www.amazon.com/Practical-Software-Metrics-Management-Improvement/dp/0137203845) when he was working with Hewlett-Packard.

FURPS+ is an acronym that outlines five key attributes for use in software quality assurance:

* **Functionality**: How well the software performs its intended tasks and how it manages security while doing so.
* **Usability**: How user-friendly the system is, how it looks and feels and how well it is documented.
* **Reliability**: How frequently the software fails, whether that be cataclysmically crashing or simply providing unexpected results.
* **Performance**: How fast and efficient the software performs, as well as how scalable the system is for future growth.
* **Supportability**: How flexible the system is to changes in components, systems, repairs, and so forth.

Each of these rough categories can be further drilled into to the most detailed degree, but with just these basic attribute categories to work with, software quality assurance engineers can begin the process of systematically testing each component (and thus, attribute) of the entire system.

## Typically Tantalizing Types of Testing

With our basic software quality attributes defined, it's also important to understand the wide range of testing methods that can be performed. Just as with our attributes example above, this is by no means an exhaustive list, but below we'll examine a few of the common test types used throughout the industry.

* **Requirements**: Before any testing can occur, it must be established what the expected requirements (or results) are of a given test. In other words, taking the established attributes, how can those attributes be systematically measured during testing?
* **Unit Testing**: Ideally a unit test is intended to test the smallest possible part of the software, down within the individual functions and methods in the source code. Therefore, unit tests are typically created (and performed) by developers rather than software quality assurance.
* **Integration Testing**: Tests which measure how well the software integrates with other which are components necessary for its functionality (such as third-party services or other, localized systems).
* **System Testing**: Typically, this is the bread and butter of the testing procedures, and covers the core functionality of the completed system in relation to the established requirements.
* **Regression Testing**: As development on a project continues, or as necessary changes to a released product are made, it is often necessary to perform testing on components that have already been tested and passed. This act of repeating tests is known as regression testing.
* **Performance Testing**: This category of tests are performed to ensure the measurable levels of performance are within acceptable standards. This might include response times on web pages, load times for desktop/mobile apps, or general system stability.
* **User Acceptance Testing**: This stage of testing is intended to ensure that end users are able to properly test and use the software. While this form of testing can be performed by public members, often it is best performed by in-house or private firms like Test.io, to ensure the system performs as expected in all manner of user scenarios.
* **Bug Tracking**: By and large, one of the most critical components during any successful software development life cycle is the ability to properly locate, track, and resolve the plethora of bugs and defects that will inevitably rear their ugly heads. Software quality assurance is the most critical department for tracking and retesting these issues.

## Learning Quality Assurance on the Job

Stepping into the world of software quality assurance has never been easier, through organizations like Test.io and the dramatic growth in the industry of [crowdtesting](https://test.io/blog/managing-crowdtesters-best-practices/).

At the most basic level, crowdsourced testing (or crowdtesting) is a means by which software testing and quality assurance is implemented with the efficiency and effectiveness of crowdsourcing and by utilizing remote cloud platforms. By giving the keys to the testing kingdom to hundreds or even thousands of individual testers around the world, crowdtesting allows for a given project to be tested, checked, then retested again by an extensive and exhaustive team of experienced personnel.

Best of all, platforms like Test.io make it easy to get your foot in the door and start down the path of a software quality assurance professional. Garnering experience through such a program allows for newbies in the industry to receive not only hands-on experience, but frequent feedback from other senior testers who have years if not decades of QA experience under their belts.

Even for projects with exceptional, automated testing already in place, the additional failsafe of crowdtesting ensures that issues that are commonly overlooked by automatic software, particularly those in the front-end, can be identified and addressed as quickly as possible.

**SOURCES**

* http://www.sqa.net/
* https://en.wikipedia.org/wiki/FURPS
* https://en.wikipedia.org/wiki/Crowdsourced\_testing