Quality Assurance

Web Development
Foundations

What is Quality Assurance?

An excerpt from the Merriam-Webster (online) dictionary:

Quality Assurance

a program for the systematic monitoring and evaluation of the various aspects of a project, service, or facility to ensure that standards of quality are being met

(*Source*: https://www.merriam-webster.com/dictionary/quality%20assurance)

Essentially, quality assurance efforts are steps we take to set a standard of quality that should be adhered to, and ensure that we objectively reach that set goal. It can take shape in a number of ways, and usually requires multiple processes working in unison to reach a result of the desired quality level.

In this lesson we will focus largely on software project quality assurance techniques and concepts, but do note that many types of teams and projects can benefit greatly from implementing quality assurance practices in their work.

Common Practices in Quality Assurance

There are many processes and practices that can be implemented to help ensure that a certain level of quality is reached; some include:

- Version Control
- Code Review
- Testing
- Software Design
- Requirements Engineering

Version Control

Version Control

Version control is an essential part of many web developer workflows. There are a variety of advantages to using a version control software. Some popular version control solutions include:

- Git
- CVS (Concurrent Versions System)
- SVN (Subversion)
- Mercurial

The idea behind these softwares is, as you are working on a project, you can set “checkpoints” or “milestones” as you work. These become different versions of the project, that you can recover at any time.

This means if something *used* to work, but doesnt now, you can track what changed and get to the bottom of it! It can also help team members track your approach to better understand your code.

Code Review

Code Review

Typically in a version-controlled development environment, it is helpful to run code reviews. The way this works goes something like so:

1. A programmer will write a new function for the program or project.
   Try to keep changes small, focused, and well-written!
2. These changes will be saved as a version, but not added to the production (launch-ready) copy of the project. This version is flagged for review by a senior member of the team.
3. The programmer and the senior team member will review this addition to the project together.
4. If there are concerns, improvements to be made, or changes, they will be addressed by the programmer at the senior team member's request, reviewed again, and ultimately approved.
5. If or when the changes are approved, they can then be added to the production copy of the project, and included in the next deployment.

Testing

Testing

Testing can take many forms when it comes to software, and there are often multiple steps involved depending on the processes selected for any given project.

There are types of testing that can be performed via programmed automated tests, and there are a variety of different approaches for user-based testing, depending on what you are hoping to evaluate. A combination may be the strongest approach, but sometimes budgetary concerns can be a factor in what is planned for the project. The larger the project, the more you'll want to ensure there are testing practices in place to keep the product easier to maintain and keep its quality high.

Let's go over a few types of testing (note there are many more, so it is good to research what is available in the industry.) Some of these terms may have overlap, or apply to other terms and strategies that we cover.

- Box
- Alpha and Beta Testing
- Accessibility Testing
- Unit Testing
- Continuous Testing
- A/B Testing

“Box” Testing

One way to describe a few major types of software testing fall under the umbrella terms provided via the “box” label(s). The two major categories here are as follows:

Black Box

This is any type of testing where the code, programming, and inner components do not need to be known.

Think of something like a typical television remote, you would be testing something you cannot see inside.

White Box

White box testing, in contrast, involves testing of the inner components of software. Also called “clear” or “glass” box testing, you are testing something with full knowledge of the code or programming inside.

This often includes programmed or automated testing methods.

Note that there are other types of box testing, like “grey box testing,” that you may want to explore too!

Alpha and Beta Testing

Alpha testing is typically done near the end of the development of a project or release thereof. Tests will be completed by the developers of the project, and/or a quality assurance / testing team of some description.

It is in this stage of testing that the goal is to find previously non-encountered bugs or shortcomings prior to final steps in the project, so that they may be addressed.

After alpha testing is completed, and any necessary updates are made to the project, beta testing may begin. This testing is usually more focused on the average user's experience with the software, and may include more testers or chosen users from outside of the development organization. Sometimes beta versions of software are released, usually in controlled quantity, to the public to receive even more feedback.

Once feedback has been collected from beta testing, any new updates or changes to the project can be scheduled and dealt with as appropriate.

Accessibility Testing

As applications on our phones, tablets, televisions, computers, and more become more and more integral to daily life it is important that we remember that everyone should have access to this incredible technology.

Consider a grocery store. It likely has a ramp, in case someone is in a wheelchair or might have trouble with steps.

When it comes to many of our devices, websites, and applications there are also steps we can take to ensure as many people as possible can enjoy the content or features they afford. It is important to run tests to ensure websites and applications are easy to use for colour-blind, visually-impaired, deaf, one-handed, or any people with other potential conditions or circumstances that may otherwise affect their use of the software.

As this is so important cases, like Dominos got involved with during 2019, have entered the spotlight.

Unit Testing

Unit testing typically involves a careful development approach in which features are broken down into distinguishable modules of code. Each essential or important piece will often have tests programmed to assess whether it is, or is not, functioning properly—each test referred to as a “test case.”

Before running the software in full, these test cases can each be run to identify damaged, incomplete, or broken functionality in the program before even experiencing it as an end-user. This becomes especially handy to have when updating or upgrading code that might replace current production functionality, as you may catch something that could have otherwise been missed.

Because test cases must be planned and programmed, it is important to take one’s time and carefully consider what should be tested and how, for the most utility.

Continuous Testing

Continuous testing often incorporates unit testing or an equivalent practice that can be automated to always run the tests at every attempt to add new code to the codebase. Every time a team member submits a completed change to their version control (or another trigger similar to this) the tests will run as an additional check to ensure that this code has not compromised any essential functionality and would be ready (or closer-to) for launch/deployment.

A/B Testing

In an A/B testing scenario, two different versions of a software feature or layout will be presented to different groups of users at random. The success rate of users in both cases will be measured to determine which is more effective in achieving the desired ends.

Software Design

Software Design

Software design, like many quality assurance topics, can find itself realized in various forms.

Typically designing software will involve a software requirements analysis, wherein the purpose and goals of the project are discovered and defined. From there the approach can be discussed and documented. Documentation may involve written plans, flowcharts, pseudo-code, storyboarding, or other team-friendly aids.

Often software frameworks or design patterns are mentioned in planning as to ensure consistency and a solid foundation for the project.

To design good software, there are many concepts, considerations, and principles that should be kept in mind depending on the type of software you plan on building. Research best practices and maintainability strategies.

Requirements Engineering

Requirements Engineering

For software development, it can be incredibly helpful to lay out and engineer requirements for any given project. This will usually mean having members of the development team meet with the client to understand their needs. Once an understanding is reached, a written or visual representation of the project might be developed to be reviewed by all involved parties. Assuming this is the case, and this documentation is approved, it is possible a more in-depth modeling of the project can occur—otherwise a formal document can be completed and finalized for the project.

Without a clear set of requirements and guidelines for a project, it can be easy to get lost and add extra or unnecessary features—or even worse, forget essential features that were meant to be a part of the product to begin with! Such documentation also helps one prevent common downfalls of developments reaching success, like scope creep.

Recommended Readings

For more terminology, examples, and exercises try:

- Patton, R. (July 2005). *Software Testing, Second Edition.* Sams.
- Mahfuz, A. S. (April 2016). *Software Quality Assurance*. Auerbach Publications.