International Journal of Trend in Scientific Research and Development (IJTSRD)
Volume 4 Issue 4, June 2020 Available Online: www.ijtsrd.com e-ISSN: 2456 – 6470

How Quality Assurance is Important in Development Life Cycle

Aayush Tiwari

Department of Computer Science Engineering and Information Technology,
Dronacharya College of Engineering, Gurgaon, Haryana, India

ABSTRACT

In recent days the requirement of quality has grown up universally. Away back, quality culture was practiced only by software industries, today it has also been embraced by companies using technology in their internal projects. The project should not be risked due to contingency and advancement that may come along the way. This is one of the precepts of software quality. Capacity development are imminent, but must be planned for another sprint, unless there is an urgent need on the part of the customer. The software architecture must adhere to the specified requirements. It can be innovative, lasting and still try to solve, not only the predicted problems, but also the unforeseen ones. It is at this point that the development team must glimpse what is really desired and not just what was asked for. One of the biggest causes of failure in software projects is lack of scope. In the eagerness to start work soon, the scope definition phase is reduced to the extreme. The result of this is a large number of corrections made during a sprint for a feature that was not properly planned.

KEYWORDS: *Quality assurance, Quality principle, Software Engineering, Setting environment etc*

How to cite this paper: Aayush Tiwari "How Quality Assurance is Important in Development Life Cycle" Published in International Journal of Trend in Scientific Research and Development (ijtsrd), ISSN: 2456-6470, Volume-4 | Issue-4, June 2020, pp.1474-1478, URL: www.ijtsrd.com/papers/ijtsrd31580.pdf

IJTSRD31580

1. INTRODUCTION:

Quality Assurance is the prepared and precise firm of exercise that assure that software processes and products conform to needs, specifications and activities. Processes include all of the activities involved in designing, developing, enhancing, and maintaining software. Products include the software, associated data, its documentation, and all supporting and reporting paperwork. QA includes the process of assuring that standards and procedures are established and are followed throughout the software development lifecycle. Standards are the established criteria to which the software products are compared. Procedures are the established criteria to which the development and control processes are compared.

Compliance with established requirements, standards, and procedures is evaluated through process monitoring, product evaluation, audits, and testing. The three mutually supportive activities involved in the software development lifecycle are management, engineering, and quality assurance. Software management is the set of activities involved in planning, controlling, and directing the software project. Software engineering is the set of activities that analyzes requirements, develops designs, writes code, and structures databases.

Quality Assurance ensures that the management and engineering efforts result in a product that meets all of its requirements. Each of the five phases of Project Delivery Lifecycle will incorporate QA activities and deliverables that off-set the risks of common project problems. This summary of the Project Delivery Lifecycle incorporates a high-level list of the QA activities and deliverables associated with each phase.

SQA always holds an important position because in developing software, IT companies use plenty of time, human resource and machines so even a minor mistake can have quite bad repercussions. If the company is not using standardize procedures even a small fault in the final product can cost a company heavily in the form of lost reputation and the broken trust of client who can try some other company to get the services he/she needs.

Software assurance is a crucial to check the errors and bugs in software. Software quality assurance (SQA) is a properly planned and well executed approach so that the developed software is of excellent quality. SQA starts from the initial starting phase of development process and involves the following proper standardize procedures till the end of development process. True purpose of SQA is to be sure that the final developed software is as per requirement of the client.

Quality assurance is performed to locate the errors or glitch in the application and to make sure that makes the customer journey as minimal as possible to eliminate the complex and unnecessary steps. Quality assurance is performed after almost every step of the development process with a certain outcome in mind-

1. During the assessment phase, QA editor submits revised and approved deliverable documents.
2. During the planning phase, the QA team establishes Standards and Procedures along with QA records for requirements. QA test matrix is also designed.
3. During the designing phase, the QA team confirms that all designs effectively meet all the requirements and identify if there are any conflicts or discrepancies.
4. During the development phase, the QA team develops a set of test cases for all the deliverable functionality for the current phase.
5. During the implementation phase, the team focuses on testing and reviewing each aspect of the system.

An efficiently carried out QA can ensure the quality of the project by fulfilling the different functionality aspect of the project.

A company creates and then develops an application. Developers are asked to implement the design.

In a traditional pattern, the designer would work for a while (hopefully there's a team that collectively refines the design,) then the developer writes all the code, and then finally hands the code to QA for testing.

Another unit of the team should be documentation of course.

A better way to do this would be to include QA at every step. Have QA sit in on the design of the next feature. They can add input that others wouldn't think about - this helps polish the feature as well as clarify testability.

The developer should be writing unit tests as the code is developed - QA should understand well what the feature is, and can be sitting with the developer to see these unit tests, which will help QA develop a test plan.

Once the developer feels the code works, QA should be right there to test the code. QA's attitude is very different from the developer. The developer tries to make the code work, and to polish it so it's user friendly. QA's role is to try to break the code, and find every rough edge and inconsistency. (And documentation should be part of this flow as well.)

Ideally development is done as a team - I favor the idea of a Scrum team. The team is the product owner, the developers, QA, and then sales and marketing, demo jocks, everyone. Ideally the entire team is part of the process - and QA and developers work together.

During the requirements phase, QA should be involved in requirements reviews: not to try to determine the nature of the product (that's up to product management), but to demand clear specifications that can be used to plan testing and, ideally, write test cases, even before any further work is done.

During the design phase, QA should continue to be involved in reviews. This is, again, not necessarily with a view towards changing the design - though if QA has domain expertise that can happen - but so that testing can be done in an intelligent and appropriately focused way. During coding, QA can be involved in code reviews (Fagan style), though in my opinion this is not usually a good use of time. More typically this is when QA will finish writing their test cases, and get developers or product management to review the test cases in turn. QA activities during testing I assume you are already familiar with.

Finally, during maintenance (following release), QA still has a role in reproducing problems reported by customers, and of course testing hotfixes and patches.
Example: When your browser crashes, it reopens all the tabs. It probably does this every time you open a tab. Does it store this info in a file? What if that file becomes corrupt? (Ignore the fact that this is a simple example and it may not matter). These are the kinds of insights you want to get by being involved with development.

You can also ask for testability by being involved. That is an outcome of understanding what you are looking for. How is data stored in the app, how do components exchange data/info., etc.

2. Importance

A. It saves your money and time

If bugs and defects are found in the early stages of development, then you are lucky to spend less money and time to fix them. The investment in time and resources pays off many times. Let's look at this graphic, which shows that rectifying bugs, not in time can lead to a dramatically increase of its cost over time.

B. Stable and Competitive Product

Surely everyone wants a successful product which runs consistently without crashing and works reliably, has no bugs and defects. QA processes and testing verify that the system meets the different requirements including, functional, performance, reliability, security, usability and so on. There are a lot of devices, browsers, and environments and the product should work properly in any of them.

C. Safety

When we create or develop something, we ask ourselves a simple question: is our product (it can be software, application, site, etc.) is secure, efficient and even trustworthy? Don't forget, that for companies and for companies' renown having a safety product is a must. You need to be certain that your software is not the source of the personal data breach and more importantly of the loss of your customers' and users' business data. QA *guarantees* you that.

D. Reputation

Quality Analyst work throughout the software development life cycle and apply different testing methodologies to make sure that your product will not receive bad reviews.

E. It helps meet clients' demands and expectations most fully

QA makes sure that the end result meets the business and user requirements. It ensures the reliability of the application and satisfaction of the user and is a secret key to draw development of the business.

F. New suggestions and views on your project

Who can know the entire product better than one who examines thoroughly all its pitfalls? QA Experts always can add something useful and breathe life to your project.

G. Scrap Reduction

Quality assurance systems identify areas that result in scrap, or products that don't meet company specifications. When the company reduces its number of defective products, it experiences scrap reduction. Scrap reduction results in savings; identification of defects early in the production process decreases the cost to the company, because fewer man-hours and materials have been used.

H. Time Efficiency

A quality assurance team can reduce the amount of inspections required in a manufacturing organization. The quality assurance team is separate from the production group, and can therefore be objective in identifying time-wasting areas during production. They also ensure that production workers don't use valuable production time to inspect or evaluate the production system.

I. Increased Customer Satisfaction

The quality assurance system improves the quality of products and services, which increases customer satisfaction. Customer satisfaction leads to repeat business, customer referrals, increased sales and profits. A quality assurance system eliminates defective products and continuously evaluates the process to improve products and services. Quality assurance can result in a consistently reliable product or service. Increased reliability in the end product results in customer satisfaction and brand loyalty. Companies with reliable quality gain a favorable reputation in the industry.

J. Improved Employee Morale

Employee morale is higher in a company using a quality assurance system, since the organization is more likely to run well, and actively seeks methods for improvement, according to the National Institute of Accountants. For example, a system of quality assurance, such as Total Quality Management, involves employees in the process of quality improvement. Employees become stakeholders in the organization and its success. Improved employee morale results in less absenteeism and turnover among workers.

3. Setting up environment

QA environment is also known as a Staging environment. It is a dedicated environment meant for testers and quality analysts to test any CR (change request) or bug fix before it gets successfully deployed in Production.

You can call it a mirror environment to a Production environment.

Workflow of making any update into any web app:

Dev → QA → UAT → Prod

Dev environment is the area for developers to fix the bug and QA or Staging is where the bug fix is tested thoroughly as well as regression testing takes place. After everything is tested and found okay! Testing team will provide a sign-off.

During this process, there are often discussions about how many environments a particular project really needs. One project may only have one QA environment while another may have four or five. Environment managers are frequently put in a position of having to ask teams to justify why they need so many environments.

This post is our contribution to this discussion. We'll offer arguments that show that there are scenarios that require a larger number of testing environments. We'll start by briefly covering the most common types of testing environments. After that, we'll give you 3 important reasons to use multiple QA environments.

Then, we summarize the arguments into a piece of general advice, while also granting that they may not be a one-size-fits-all solution. To wrap-up, we cover the brave new world of new hybrid cloud solutions, and how that can affect the QA strategy of your organization.

- **Development environment**. Used by the developers. In practice, this environment consists of the developers' machines themselves. They use this environment to Unit test their code before it gets to the next stage.
- **QA/Testing Environment.** This environment is used by testers, QA analysts or other testing professionals to perform many forms of functional and non-functional testing, such as end-to-end testing, load testing, integration testing, and more.
- **Staging environment.** This is essentially a copy of the production environment. It's meant to be as close as possible to production, so the team can verify if the application will behave correctly after its deployment.

4. Reasons to use more than one QA environment

A. Teams Are Working on Parallel Development Efforts

If a project has regular releases there's a good chance that when a development team is finished with a feature, a QA team takes over to validate that feature. During that QA

process, development teams often want to move on to the next feature. In these scenarios having two QA environments make sense as features can be delayed and releases will have to be serialized if a QA environment is "tied up."

B. Long-Term Feature Releases Need to Be Developed While Short-Term Bug Fixes Are Qualified and Staged

If you have a team working on a series of larger, multi-month development stories to launch a new product these efforts almost always require a dedicated environment. These are QA efforts that take months, and require customizations to databases that cannot ship to production. If you don't have an isolated system for these longer-term initiatives you will be unable to fix bugs as they are identified in a production system.

C. Systems That Rely on Services

If you develop code that relies on back-end services that are also being modified by independent development teams these teams may require multiple environments that are configured to connect to the appropriate testing service. Service-oriented architectures and micro services because a combinatorial increase in the number of environments required to perform end-to-end testing.

5. Principles

It is important that you achieve optimum test results while conducting software testing without deviating from the goal. But how you determine that you are following the right strategy for testing? For that, you need to stick to some basic testing principles. Here are the common seven testing principles that are widely practiced in the software industry.

To understand this, consider a scenario where you are moving a file from folder A to Folder B.

Think of all the possible ways you can test this.

Apart from the usual scenarios, you can also test the following conditions

- Trying to move the file when it is Open
- You do not have the security rights to paste the file in Folder B
- Folder B is on a shared drive and storage capacity is full.
- Folder B already has a file with the same name, in fact, the list is endless
- Or suppose you have 15 input fields to test, each having 5 possible values, the number of combinations to be tested would be 5^15

If you were to test the entire possible combinations project EXECUTION TIME & COSTS would rise exponentially. We need certain principles and strategies to optimize the testing effort

Here are the 7 Principles:

A. Exhaustive testing is not possible

Yes! Exhaustive testing is not possible. Instead, we need the optimal amount of testing based on the risk assessment of the application.

B. Defect Clustering

Defect clustering which states that a small number of modules contain most of the defects detected. This is the application of the Pareto Principle to software testing: approximately 80% of the problems are found in 20% of the modules.

By experience, you can identify such risky modules. But this approach has its own problems

If the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs.

C. Pesticide Paradox

Repetitive use of the same pesticide mix to eradicate insects during farming will over time lead to the insects developing resistance to the pesticide Thereby ineffective of pesticides on insects. The same applies to software testing. If the same set of repetitive tests are conducted, the method will be useless for discovering new defects.

To overcome this, the test cases need to be regularly reviewed & revised, adding new & different test cases to help find more defects.

Testers cannot simply depend on existing test techniques. He must look out continually to improve the existing methods to make testing more effective. But even after all this sweat & hard work in testing, you can never claim your product is bug-free. To drive home this point, let's see this video of the public launch of Windows 98

Let think a company like MICROSOFT would not have tested their OS thoroughly & would risk their reputation just to see their OS crashing during its public launch!

D. Testing shows a presence of defects

Hence, testing principle states that - Testing talks about the presence of defects and don't talk about the absence of defects. I.e. Software Testing reduces the probability of undiscovered defects remaining in the software but even if no defects are found, it is not a proof of correctness.

But what if, you work extra hard, taking all precautions & make your software product 99% bug-free. And the software does not meet the needs & requirements of the clients.

This leads us to our next principle, which states that- Absence of Error

E. Absence of Error - fallacy

It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement. Software testing is not mere finding defects, but also to check that software addresses the business needs. The absence of Error is a Fallacy i.e. finding and fixing defects does not help if the system build is unusable and does not fulfill the user's needs & requirements.

To solve this problem, the next principle of testing states that early Testing

F. Early Testing

Early Testing - Testing should start as early as possible in the Software Development Life Cycle. So that any defects in the requirements or design phase are captured in early stages. It is much cheaper to fix a Defect in the early stages of testing.

But how early one should start testing? It is recommended that you start finding the bug the moment the requirements are defined. More on this principle in a later training tutorial.

G. Testing is context dependent

Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application. All the developed software's are not identical. You might use a different approach, methodologies, techniques, and types of testing depending upon the application type. For instance testing, any POS system at a retail store will be different than testing an ATM machine.

Principle 1	Testing shows presence of defects
Principle 2	Exhaustive testing is impossible
Principle 3	Early Testing
Principle 4	Defect Clustering
Principle 5	Pesticide Paradox
Principle 6	Testing is context dependent
Principle 7	Absence of errors - fallacy

6. How to do Quality Assurance: Complete Process

These above steps are repeated to ensure that processes followed in the organization are evaluated and improved on a periodic basis. Let's look into the above steps in detail -

- Planning Organization should plan and establish the process related objectives and determine the processes that are required to deliver a high-Quality end product.
- Do - Development and testing of Processes and also "do" changes in the processes
- Check - Monitoring of processes, modify the processes, and check whether it meets the predetermined objectives
- Act - Implement actions that are necessary to achieve improvements in the processes

7. ISO Standards

ISO 9000

This standard was first established in 1987, and it is related to Quality Management Systems. This helps the organization ensure quality to their customers and other stakeholders. An organization who wishes to be certified as ISO 9000 is audited based on their functions, products, services and their processes. The main objective is to review and verify whether the organization is following the process as expected and check whether existing processes need improvement.

This certification helps -

- Increase the profit of the organization
- Improves Domestic and International trade
- Reduces waste and increase the productivity of the employees
- Provide Excellent customer satisfaction

8. Conclusion:

Software Quality Assurance is about engineering process that ensures quality It. Involves activities related to the implementation of processes, procedures, and standards. Example – Audits Training

Quality Assurance is to check whether the product developed is fit for use. For that, Organization should have processes and standards to be followed which need to be improved on a periodic basis. It concentrates mainly on the quality of product/service that we are providing to the customers during or after implementation of software.

References:

[1] http://www.onestopsoftwaretesting.com/introduction-and-importance-of-software-testing-in-sdlc/

[2] https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-01sc-introduction-to-electrical-engineering-and-computer-science-i-spring-2011/unit-1-software-engineering/

[3] https://www.google.com/search?q=software+engineering+geeksforgeeks&oq=Software+engineering+gee&aqs=chrome.0.0j69i57j0l6.8927j0j7&sourceid=chrome&ie=U TF-8

[4] https://www.guru99.com/software-testing.html

[5] https://devblogs.microsoft.com/premier-developer/the-importance-of-quality-assurance-in-the-development-life-cycle/