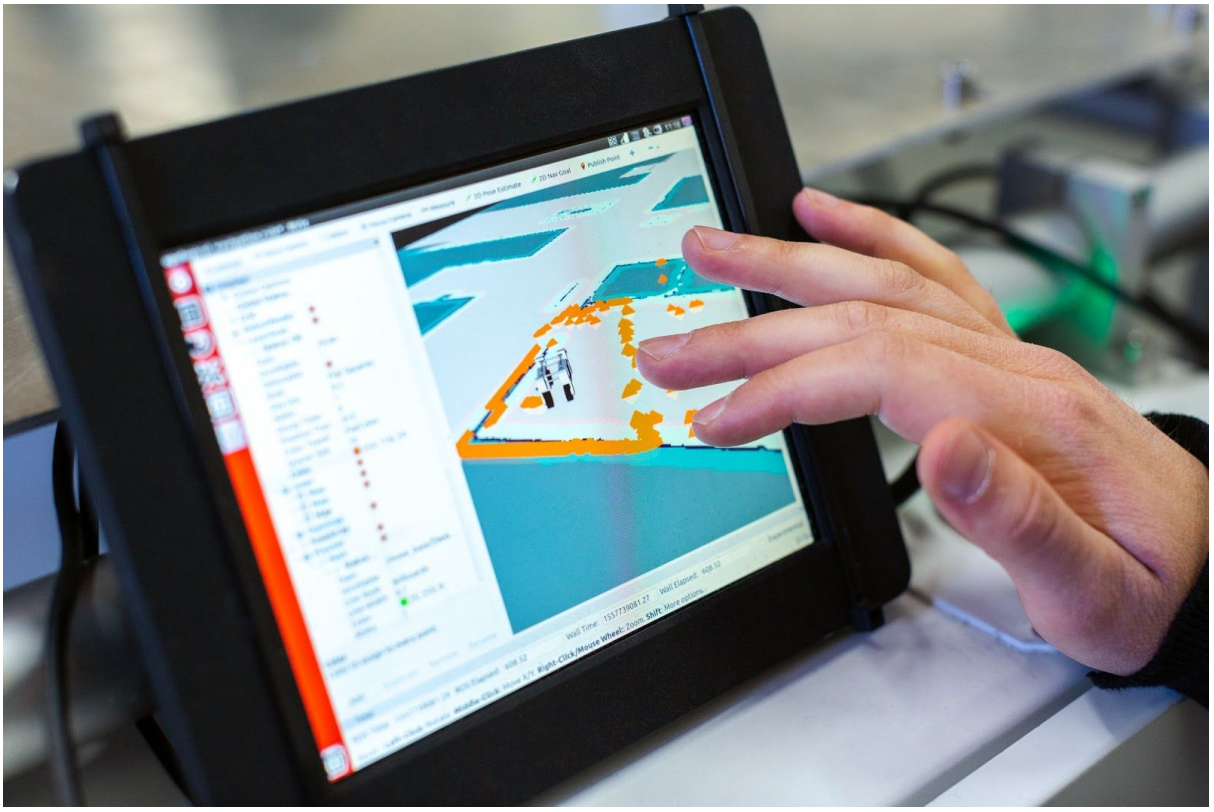
Software testing 3.0 - The ongoing evolution of software testing



What is software testing?

Software testing is a process by which developers and testers (or the companies that they work for) check whether a software product is matching the expected requirements of the client and whether the software is free of defects. Software testing involves the execution of software/system components by the usage of manual or automated tools to analyse multiple characteristics that may be of interest to the developer or to the client. The main objective of software testing is to make sure that the team identifies the various loopholes gaps or missing requirements in the software with relation to the actual requirements of the client.

There are several people who prefer referring to software testing with the terms “White Box and Black Box Testing.” In other words, software testing is a process that is performed after the development of the software to find out whether the software is ideal and whether it is developed well enough to perform ideally in testing environments.

Why is software testing important?

Software Testing is to identify any bugs or errors in the software. Software testing can also make sure that these errors and bugs can be solved and removed before the software

product is delivered to the client. There are several different aspects that make a software product ideal for development and subsequent usage. They are:

- Reliability
- Security
- High performance
- Time-effectiveness
- Cost-effectiveness
- Customer satisfaction

What are the benefits of software testing?

Software testing is such a fundamental concept in the field of software development that the core advantages of the process are already well-known by so many people in this area. For those that don't know the core advantages of software testing apart from the obvious, here they are:

- **Cost effectiveness -** This is one of the most important benefits that come with software testing. When you test your software intensively before you roll it out to the public or deliver it to your clients, you make sure that it works well. Hence, you don't have to spend tons of money on market research or subsequent updates that contain debugging programs or software updates that contain bug-fixes. Also, it is a fact that if the bugs in the software are caught in the early stages of the development, it takes a significantly less amount of money to remove them from the software. Hence, this way, you can save a lot of money by performing software tests more frequently and as early as possible.
- **Security -** This is one of the most important benefits that come with the entire process of software testing. Security is the number one priority for app developers who make apps that are used by huge corporations on a day-to-day basis. Hence, making software that has to bear the load of terabytes of data every single day is a huge responsibility to bear. Hence, this is the most vulnerable and sensitive benefit of software testing. Since people are looking for software that is already functional and secure, software testing reduces the risk of finding errors later on and having to remove them at the expense of the trust of the client and the extensive resources that updates usually take up.
- **Enhancement in Product quality -** When you test your software before rolling it out to your customers, you can ensure that it will behave exactly as you intend it to. Hence, when your client gets the software, the quality of the software is enhanced thanks to the extensive testing you performed before rolling it out. There is also another aspect to this. When you put out your software after extensive compatibility testing, you can assure the client that the software is fit to be run in different kinds of environments too. Hence, this increases the viability and the compatibility of the software that you're giving out, making it more enhanced and lucid.
- **Customer Satisfaction -** The main objective of any software product is to satisfy the clients of the customers you're aiming it for. Hence, UI/UX testing makes sure that the kind of customers and clients you're catering to love the product that you have designed. UI/UX Testing ensures that your clients, customers and target audience get the best user experience, just like you intended.

The evolution of software testing

Over the decades, software programming has evolved drastically. As a consequence of that evolution, testing has also made strides in the right direction and has become an integral part of software development. Testing, too, has gone through a series of changes through evolution to become an important and indivisible part of the development of software.

First, before software testing became a study and discipline of its own, debugging was known as testing. The phase of debugging the software from loopholes in the code and glitches that it was prone to was known as software testing. In 1957, however, software testing got attributed the recognition of being an individual identity and was treated as a separate activity from the debugging process. It was till the late 1970s that testing was seen as an exercise that helped to make sure that the software product that was designed indeed worked as per the specified requirements of the client or the customers. The concept of software testing was then extended to the realm where testers found the errors, apart from making sure that the software functioned properly. In the 1980s, the amount of testing that was conducted on software became an indicator of the quality of the software. With this, software testing and its different types gained more importance and were treated as a clearly defined and managed method that encompassed the software development life cycle. By the mid-'90s, the testing process had become an entirely different phase where people became aware of its merits.

In this section, we have taken the liberty to classify the different phases of software testing evolution into the different eras for a better understanding of how events unfolded, and why they unfolded the way they did.

The Era of Testers

Development and testing were considered to be mutually independent activities during this era. Once the software product's development was ready, the software was sent to the testing team for complete verification. These testers of the code were hardly involved during the analysis phase and had very restricted interactions with the stakeholders of the development process of the software product. The testers were heavily dependent on the information and knowledge that was passed on to them through the documentation that was done during design and development. The severe lack of insight into the requirements and expectations of the customers they made the product for led to restricted testing strategies used by the team responsible for testing. The testers would hence, develop a test plan that was entirely based on their understanding of the documentation. They would also go ahead and test the software in an ad-hoc manner.

The Era of Exploration and Manual Testing

The software development and testing era that began in the late '90s saw the advent of several methodologies like agile testing, exploratory testing, etc. During this time, testing was done manually with the usage of elaborate test cases and test plans.

Exploratory testing gave testers and their teams the freedom to test and break software in unique and distinctive ways by venturing the software within the gamut of testing charters.

More comprehensive ways of testing were needed to support the expansive and intensive growth of the software development process in this era. The approach that was popularly adopted by agile testing did a lot to achieve this goal. It was these iterative tests that eventually made the way for automation of tests that were repetitive in nature; in fact, *too* repetitive for humans to perform on a daily basis.

The Automation Era

There came multiple new approaches to software testing with the new millennium's advent. These approaches gave a complete makeover to software testing and revolutionized the way people looked at the process. Testing now became a very integral part of SDLC at every step. Quality assurance and control of and in the software product in every phase gained importance.

Automation took software testing to an entirely different level. With the existence of a wide range of automated testing frameworks, the testers became empowered to carry out their testing tasks with improved efficiency. Automation also greatly helped in carrying out regression and sanity testing with efficiency, accuracy and speed.

It was this era that also saw the need for scaling up the testing process for the entire software industry. Concepts like crowdsourcing and cloud testing helped businesses in testing products at a faster and more efficient pace and manner along with significantly lesser investments in infrastructure.

The Era of Continuous Testing

The business dynamics started changing vigorously and customers now anticipated an intermediate working model of the end product that was being designed. Thus, the demand for software releases that were frequent and efficient went up drastically.

There were several factors that improved the speed of testing across multiple platforms. They were an improved network infrastructure that provided high connectivity and improved speed of deployment and testing. This also helped in increasing the frequency of deliveries which basically added to more testing work.

New and improved concepts like Continuous Integration and Continuous Deployment became very popular in this era. Continuous testing also became very popular during this time due to the rise in popularity of such concepts in the software industry.

Shorter delivery cycles were a direct consequence of the rise of Developer Operations and CI/CD. The need of the hour was the careful assessment and analysis of risk. Risk assessment and handling had to be done in all stages of the software development process. Continuous testing became the best way to manage these bugs efficiently before the software was released. This led to the further rise in the importance of software testing and how people in the software development industry regarded it.

There came more and more business stakeholders that demanded the intermediate release of softwares within tight deadlines, and making sure that no compromises were made with the quality of the end product that was rolled out to the consumer or the client. To keep in touch with these demands and make sure that they were met completely, continuous testing was evolved. Continuous testing became more efficient and that was when artificial intelligence made its entrance there.

The Era of Artificial Intelligence

In simple words, Artificial Intelligence can be defined as the machine's ability to imitate human behavior by self learning. The algorithms for Artificial Intelligence are based on the insolent and predictive analysis of data. This also implies that Artificial Intelligence testing depends heavily on data.

There are many testing tools that are powered by Artificial Intelligence that help with unit testing, API testing, UI testing, etc. Visual testing is a classic example of the usage of Artificial Intelligence in testing.

The evolution of software testing with reference to the different types of testing

Here are some of the different types of software testing evolutions that have happened with reference to the specific types of testings involved. Read on to find out more.

DevOps and Continuous Integration Testing (CI)

A DevOps team is one that consists of incessant and constant communication along with iterative and repetitive movements. With this development in software testing, automation has become very critical. With the usage of Continuous Integration (CI), all developer copies are merged into a shared mainline for testing multiple times a day. Continuous Integration testing is applied to a lean workflow using tools that are used for automation. These tools watch out for changes in code and run the applicable tests on them immediately.

Agile Software Testing

The role of the tester in an Agile team is not just about the traditional tester role of logging and identifying defects in the software. It extends to the development team itself. The tester is looped into all the parts, planning and activities of the development process and is in close coordination with the product owner at all times. In Agile, testing is an integral part of the development lifecycle, right from the beginning of the process. This is to ensure that quality is built in from the start itself and there are no bugs that need to be corrected. By the end of each sprint (based on the test results), the consumers can see the actual product and provide valuable and related feedback which can later be used for improvements. Manual and automatic testing can both be done for this method.

Behavior-Driven Development (BDD)

Behavior-driven development (BDD) emerged from test-driven development (TDD), a practice wherein you write a test that deliberately and intentionally fails before writing new code that is actually required and is functional. In the same way, with BDD, the test is written first, what do you want people to do (or not to do), and then code is developed in such a way to achieve that outcome through the software in the end. These tests are purposefully written in plain, language that is descriptive instead of code. This approach brings the business team a lot of benefits. For instance, it brings the business team closer to the testing, designing tests around the actual defined behavior and the overall end outcomes of the same. This practice requires intensive collaboration between various business stakeholders, business analysts, Quality Assurance teams, and different teams of developers. Multiple frameworks including but not limited to Cucumber or JBehave can be also used as the tool that acts as a functional bridge between business and technical language that is often used in the software industry.

Cloud-Based Performance Testing

The advantage of cloud-based testing is the inherent ability to test applications at a large scale. Testers are not confined to the size and scope of the development environment that they are working in. We can simulate real-world usage for a lot better experience of performance testing. Cloud-based testing also allows for testing from anywhere in the world, which provides an inside look into how users in different geographical locations can access and interact with the application through the cloud. This technology, however, is not new. Cloud-based systems have been around for a long time. Cloud-based performance testing allows testers the ability to use the simulation of the real peak load before launching any new application. This gives them a huge edge over the competition.

Web Services Testing

Web Services can be defined as a mechanism that two applications or machines use to communicate and exchange data with each other. As apps seldom operate in a vacuum, it is very pertinent to also test the web service to make sure that there are acceptable performance metrics between dependent applications. There are some key things to look for. They include response time, response data validation, scalability and transaction

volume, and security. Ultimately, many believe that automation is key to effective web services testing and as such there are a number of tools on the market that are using protocols like using SOAP, REST, and HTTP protocols. The main objective for web services testing is to identify whether the application is getting correct data from other systems within the specified time frame.

Mobile Applications Testing

Mobile application testing, contrary to popular opinion, is not testing on devices, rather it is testing applications and how they perform on mobile devices. The app itself may not be mobile-based or made to work on mobile devices, but it does need to be accessed on mobile devices via mobile browsers by using native apps. The primary focus of this testing is:

- Installation
- Performance
- Functionality
- Security
- Network usage
- Battery usage

All of this can also be completely automated or performed manually with real or emulated devices. Thus, this is a really versatile and popular way of testing applications.

Conclusion

As the Information Technology industry's infrastructure and software development has evolved so much, so have the various options that are available to testers for application testing. It has already been popularly established across the industry that there is no one right way to test. It has also been popularly agreed among testers that the tactics and methodology depend heavily on the development style and cycle. Not only that, it also depends on the use case for the application. With that said, Continuous Integration Testing is standing apart from the competition to be the leading approach for software testing. This is because it provides the benefits of early testing for any changes and allows for fixes very early during the development cycle.

There are many benefits of software testing. Since the concept was first thought of, there were many people who were a little skeptical of it. But as the software industry made strides in the positive direction, so did software testing. Slowly and steadily, more and more developers and companies started to realize the importance of software testing. There are many advantages to testing your software early. We have given you several examples where we say why software testing is much more profitable and cost-efficient when it is performed frequently and in the early stages of the software development process. Furthermore, before rolling out a software product to your customers (the end user) or to the client who has provided you with the requirement, it is very important to perform tests. Tests like compatibility testing are very important. This makes sure that the client recipes a software that is not only made according to the requirements, but also made to work in all compatible devices. This increases the credibility of your software in a way. It also makes your software more compatible and increases the usage. However, compatibility testing is just one example. There are several other important tests too.

Hence, this article gives you valuable insights about the evolution of software testing, including the history of software testing. We have tried to make this document holistic. We hope that you find it valuable to your knowledge. Good luck!