iamneo

Introduction to Software Testing

Introduction to Software Testing

Software testing is the act of examining the artifacts and the behavior of the software under test by validation and verification.

What is Software Testing

Several definitions:

“Testing is the process of establishing confidence that a program or system does what it is supposed to.”

by Hetzel 1973

“Testing is the process of executing a program or system with the intent of finding errors.”

by Myers 1979

“Testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results.”

by Hetzel 1983

IEEE Definition

Software testing is the process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs) and to evaluate the features of the software item

What is Software Testing

- One of very important software development phases
- A software process based on well-defined software quality control and testing standards, testing methods, strategy, test criteria, and tools.
- Engineers perform all types of software testing activities to perform a software test process.
- The last quality checking point for software on its production line

Principles of Software Testing

- Principle #1: Complete testing is impossible.
- Principle #2: Software testing is not simple.

  Quality testing requires testers to understand a system/product completely

  Quality testing needs adequate test set, and efficient testing methods

  A very tight schedule and lack of test tools.
- Principle #3: Testing is risk-based.
- Principle #4: Testing must be planned.
- Principle #5: Testing requires independence.

Principles of Software Testing

- Principle #6: Quality software testing depends on:

  Good understanding of software products and related domain application

  Cost-effective testing methodology, coverage, test methods, and tools.

  Good engineers with creativity, and solid software testing experience

Principles of Software Testing

Principle 1: Testing shows the presence of bugs (but a failure to find bugs does not necessarily mean that none exist). Tests must be designed to find as many bugs as possible, and since it is assumed that every product initially has bugs, a test that identifies bugs is better than one that finds none.

Principle 2: Exhaustive testing is impossible because there are usually too many variables that come into play. Therefore, testers must focus efforts on only the most critical priorities and risks.

Principle 3: Testing should be done as early as possible after the product or application has been created, in order to fix issues as fast as possible. Errors identified later in the process tend to be more expensive to mitigate.

Principle 4: Use defect clustering, as software problems tend to cluster around narrow areas or functions. By identifying and focusing on these clusters, testers can efficiently test the sensitive areas while concurrently testing the remaining “non-sensitive” areas.

Principle 5: Use a variety of tests and techniques to expose a range of defects across different areas of the product. Avoid using the same set of tests over and over on the same product or application, because this will reduce the range of bugs you will find.

Principles of Software Testing

Principle 6: The same tests should not be applied across the board because different software products have varying requirements, functions and purposes. For example, a website should be tested differently than a company Intranet site.

Principle 7: Don’t buy into the absence of errors fallacy. In other words, a test that finds no errors is different than concluding that the software is error-free. It should be assumed that all software contains some faults, even if said faults are hidden.

Software and Scope for Error, Fault, & Failure

Error : A static defect in the software Incorrect instructions, missing instructions, extra instructions

Fault: An incorrect internal state that is the manifestation of some fault

Failure : External, incorrect behaviour with respect to the requirements or other description of the expected behaviour

Testing Objectives

The Major Objectives of Software Testing:

- Uncover as many as errors (or bugs) as possible in a given timeline.

- Demonstrate a given software product matching its requirement specifications.

- Validate the quality of a software testing using the minimum cost and efforts.

- Generate high quality test cases, perform effective tests, and issue correct and helpful problem reports.

Testing Objectives

Major goals:

Uncover the errors (defects) in the software, including errors in:

- requirements from requirement analysis
- design documented in design specifications
- coding (implementation)
- system resources and system environment
- hardware problems and their interfaces to software

Verification & Validation

Validation:

Process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements

Validation: Are we building the right product?

Verification:

Process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase

Verification: Are we building the product right?

Verification & Validation

Verification

- Are you building it right?
- Ensure that the software system meets all the functionality.
- Verification takes place first and includes the checking for documentation, code etc.
- Done by developers.
- Have static activities as it includes the reviews, walkthroughs, and inspections to verify that software is correct or not.

Validation

- Are you building the right thing?
- Ensure that functionalities meet the intended behaviour.
- Validation occurs after verification and mainly involves the checking of the overall product.
- Done by Testers.
- Have dynamic activities as it includes executing the software against the requirements.

Software Testing Limits

- "Complete" or "100%" testing is not possible. There are always 3 lacks – resources, time, money
- Due to the testing time limit, it is impossible to achieve total confidence.
- We can never be sure the specifications are 100% correct.
- We can never be certain that a testing system (or tool) is correct.
- No testing tools can copy with every software program.
- Tester engineers never be sure that they completely understand a software product.
- We never have enough resources to perform software testing.
- We can never be certain that we achieve 100% adequate software testing.

Software Testing Myths

- We can test a program completely. In other words, we test a program exhaustively.
- We can find all program errors as long as test engineers do a good job.
- We can test a program by trying all possible inputs and states of a program.
- A good test suite must include a great number of test cases.
- Good test cases always are complicated ones.
- Software test automation can replace test engineers to perform good software testing.
- Software testing is simple and easy. Anyone can do it. No training is needed.

Software testing as a creative and lucrative career

Today, software testing is perceived as a solid and lucrative career especially for undergraduates and fresh graduates looking to expand their knowledge, build an exciting and creative career, and at the same, get paid well.

You can start out as a test engineer, and then later on move on as a senior test engineer, test manager, QA lead, and finally, a test consultant. But in order for you to advance in this career, you must equip yourself with the needed skills and knowledge for you to be valuable to any company.

Does Software Testing involve Scripting?

Good testers have critical thinking, analytical, and investigative skills. They understand risk and have a deep understanding where bugs tend to hide. They have excellent communication skills.

Test automation is inherently a programming activity. Anyone tasked with automating tests should know how to program (Scripting).

Market Demand in Software Testing

The market is witnessing increasing demand for testing software from companies because the quality of applications is highly dependent upon testing.

Hence, to avoid poor execution, slow test turnaround or excessive costs in the software's lifecycle, companies are relying more on testing services, thereby increasing the adoption of the outsourced software testing services.

New Trends in Software Testing

Shift Left Testing: is a better way of integrating the quality assurance (QA) and development parts of a software project. By linking these two functions at lower levels of management, you can expand your testing program while reducing manpower and equipment needs.

Testing metrics and automation: Test life cycle management, automation technologies, and new dashboards for testing metrics enhance traceability, efficiencies and the return on investment (ROI).

New Trends in Software Testing

Domain-based testing: Point / platform-based solutions that combine core business processes with advanced testing frameworks ensure superior testing.

Non-functional testing: Non-functional validation services and solutions to test usability, accessibility and performance that enhance the customer experience are gaining importance.

New frameworks: New models are required to address challenges in testing in the area of emerging technologies such as service-oriented architecture (SOA) and cloud computing.

New Trends in Software Testing

ERP Testing: Specialized testing skills and methodologies are imperative for a smooth rollout and upgrade of ERP packages.

Risk Base Testing: is a type of software testing that functions as an organizational principle used to prioritize the tests of features and functions in software, based on the risk of failure, the function of their importance and likelihood or impact of failure.

Cloud Testing: is a form of software testing in which web applications use cloud computing environments (a "cloud") to simulate real-world user traffic.

New Trends in Software Testing

Business Process Testing (BPT): introduces the concept of reusable business components for test design. This construct drastically reduces your test maintenance and improves efficiency for test creation.

Service base Testing: Service-Oriented Architecture (SOA) is a way of designing, developing, deploying, and managing enterprise systems where business needs and technical solutions are closely aligned. SOA offers a number of potential benefits, such as cost-efficiency and agility.

Mobile application Testing: is a process by which application software developed for hand held mobile devices is tested for its functionality, usability and consistency.

Roles in Software Testing