ZUCI SYSTEMS

THE ULTIMATE GUIDE TO SOFTWARE TESTING

Learn the essential turn around strategies, tips, and tactics of software testing to deliver the software to user with confidence.

TOPICS THIS GUIDE WILL COVER:

Software Testing – the definition

Software testing is the process of examining the piece of software and providing stakeholders with information about the quality of the software product or a service with a team of testers. It is one of the important stages in a software development lifecycle. Typically, it starts once after the development phase, whereas in modern agile approach, requirements, programming and testing are done parallelly. The software test processes and techniques are helpful in reducing risks associated with software quality and provides confidence in delivering the software to users.

Objectives

- The main objective of software testing is to prove the existence of bugs in the software, and not their absence.
- Providing stakeholders with the information on the quality of the product
- Early detection and prevention of defects
- Verify and validate if the product satisfies user requirements and works as desired
- Gain customer confidence by providing them with feedback on the quality of software

Software testing methodologies

Software testing methodologies involve the application of various strategies and approaches to ensure the application under test looks and functions as intended. The various testing methodologies include testing right from testing in units, integration testing, system testing and carrying out security, performance and usability testing for the non-functionality features of the application.

Types of software testing

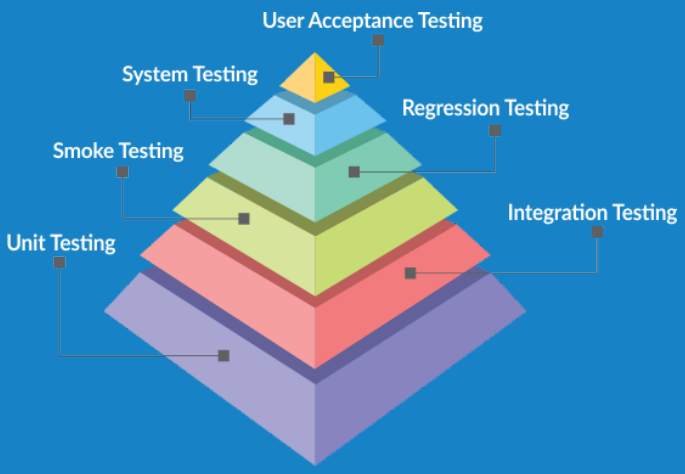
The main objective of different software testing methodologies in the SDLC is to ensure that the software can operate well across multiple environments and platforms. On broader level, we can categorize them into functional testing & non-functional testing. Functional testing guarantees the functionality/ business requirements of the software and non-functional testing deals with the operational aspects of the software like usability, performance and security.

Functional Testing

Functional testing is done to verify and validate the basic functionality and conformance of the software to the stated requirements. It is considered as a black box testing since the tester will

be testing the functionality of the software without delving into the internal structure of it, which remains a black box to the tester, After the functional testing is done, testers will log all the test reports and feed them back to the development team to fix the issues found.

Let's see the different functionality testing types and tools,



- Unit Testing
- Integration Testing
- Smoke Testing
- System Testing
- Regression Testing
- User Acceptance Testing

1. Unit Testing

Unit testing is the first level of testing process that ensures each individual units of software at the code level functions as desired. Developers writing the code will carry out these tests in test driven environment before it's passed over to the test team. It is a white box testing technique since the developer who runs the test knows the internal details of the software being tested. Unit testing can be performed manually but automating the process will fasten the delivery cycle, increase the test coverage and find defects early in the SDLC. Automated functionality testing tools help developers and testers to shift-left and fix issues early in the testing process than bugs being skipped and discovered in the later stages, costing 3X to fix them altogether.

2. Integration Testing

Each units of the software need to be integrated with each other in the form of commands or DB calls to perform the desired action, integration testing ensures these integrations between the units are flawless and whole units of the software operates as desired. Integration testing is run by developers or testers usually driven by user scenarios, either manually or in automated fashion.

3. Smoke Testing

Smoke testing is usually performed in the initial unstable builds or whenever a new build is released by the developers. This objective of smoke testing is to ensure that the critical functionalities of the application are working fine and to reject a new/broken build that affects the major functionalities of the application.

4. System Testing

System testing is a black box testing method used to evaluate the functionality of the previously completed integrated systems and to ensure, the software meets its specified requirements. It verifies the end to end functionality of the software by interfacing the software and hardware components of the entire system and is usually carried out the separate test team before pushing the product to production.

5. Regression Testing

Regression testing ensures that last modified code/code fixes/functionality enhancements do not introduce defects in the code and also ensures that previously working functionalities do not get affected with new fixes. The objective of regression testing approach is to guarentee the outcomes of these enhancements do not cause any unintended impact to the existing quality of the applications.

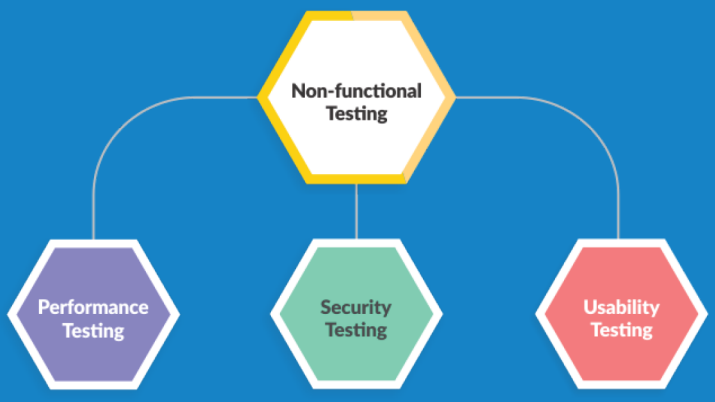
6. User Acceptance Testing

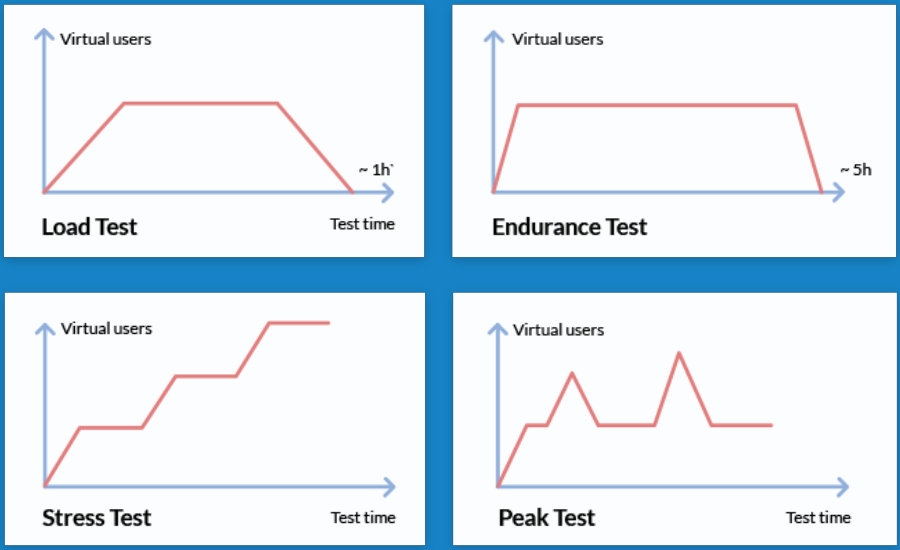
User Acceptance testing is the final phase of the functionality testing and is performed to guarantee whether or not the final product is set for release. The objective of this testing is to ensure that the final piece of software meets all the business requirements and end user's needs. It is to be run both internally by test team and externally in the hands of end user to address any functionality concerns of the product.

Non-functional Testing Types

1. Performance Testing

The goal of the performance testing is to determine the performance of the application while subjecting it to real world conditions. Various performance parameters like response time, scalability, stability and efficiency of the app in user-like conditions are monitored. The components of performance testing are,





Load Testing – process of putting increased number of simulated loads on the application and monitor the performance of the app is such real-world conditions

Stress Testing – takes load testing a step higher and is used to measure the performance of the app beyond it's acceptable load. The goal of the testing is to overload the app beyond any realistic or unrealistic load scenarios until the point it breaks so as to find the failure point of the software.

Spike Testing – is a branch of load testing used to monitor how well the software performs in case of an increased or decreased load over varying amounts of time.

Endurance Testing – also called soak testing. It is an extension of load/stress testing (applying significant amount of load over extended amounts of time) used to monitor how well the system endures under sustained use.

2. Security Testing

Security threats are real and is happening, with the rising cyber-attacks and compromising security of the software across industries, a certified security testing team is the need of the hour. The main objective of the testing to find gaps in security that risk result in unauthorized access and loss/theft of confidential information by breaking the application's firewall. Various types of security testing include,

- Application security tests (AST)
- Network penetration test
- Security vulnerabilities assessment

3. Usability Testing

Usability testing measures the usability or ease of use of the application from the end user's point of view and is often performed at the later stages of SDLC. The goal of the testing is to find the visible design and aesthetics flaws in an application and check if or not application meet the desired workflow for various process. It is an important technique to measure the user experience of the application before it can reach the hands of users. There are different subsets of testing usability, each of which aimed at verifying 5 basic areas of usability:

- Accessibility
- Identity
- Content
- Navigation
- Portability

Manual Testing & Automated Testing

Based on the test effort level by humans/machines, we can categorize testing into manual and automated testing. Test teams follow a step by step testing process to obtain the most out of testing, manual or automated.

Testing process:

Step 1 Test plan — Step 2 Test design — Step 3 Test creation & execution — Step 4 Log test results

1. Manual Testing

As the name suggests, testing is carried out manually by a team of testers. They perform the step by step testing process I.e. follow test steps in the test case, providing inputs on the application and finally validating test results. Since it involves manual testers directly involved, the amount of time and efforts increase largely for a complex test suites. However, manual testing is indispensable for scenarios best suited for human eyes to find bugs where automated testing is not possible. Example, Usability Testing – which is done from end user's view point for measuring the user friendliness/usability of the app.

2. Automated Testing

Automated testing involves automation tools and test scripts written for automation to run the test cases automatically, without manual intervention. These tools run the test cases, record the test failure/pass results and logs defects in defect management tools. All types of functional & non- functional, time consuming, repetitive tests like regression testing are best candidates for test automation. A well framed test automation strategy helps to increase test coverage, release frequency and yield a significant return on investment, by saving the time and manual efforts.

Testing Tools

Here is the list of widely used and most popular testing tools across companies,

1. Test Automation Tools

Selenium	Appium	Ranorex
TestComplete	SoapUI	Protractor
Rest-assured	Postman	HP UFT
FitNesse	Katalon	

2. Non-functional Test Tools

Apache JMeter	BURPSUITE	NeoLoad
OWASP ZAP	WEBLOAD	IBM Performance Tester
dynatrace	LoadUI	New Relic

3. Defect & Test Management Tools

JIRA	Trac	TestLink
Bugzilla	Mantis	TestRail
REDMINE	hp Quality Center	Zyphyr

4. CI/CD & Cloud Tools

Jenkins	Maven	GitLab
Bamboo	BrowserStack	SAUCELABS
Gradle	Blazemeter	Bitbucket
go	Tomcat	

Quality Engineering

When it comes to software quality, customers are relentless in their demand. Disruptive technologies and a growing digital environment have only made the scenario more challenging. A Forrester research reveals that 78% of enterprises regard quality and speed as the key factor contributing to the overall project success. In the face of such challenges, traditional quality assurance models will not suffice.

Businesses have to apply Agile methodologies, cognitive quality automation and AI to offer comprehensive Quality Engineering without increasing costs or software release cycles.

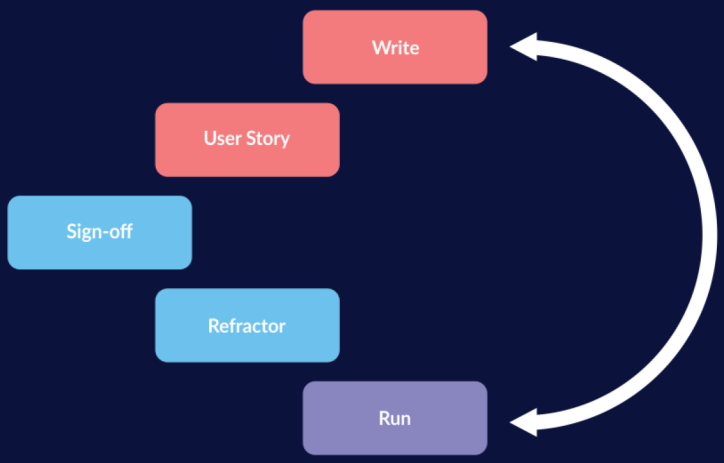
A full spectrum of Quality Engineering services spanning methodologies like Agile, Iterative and Waterfall include,

- Acceptance Test driven development
- Behavior driven testing
- Lean/Agile Testing
- DevOps Testing

Acceptance Test Driven Development

Acceptance test driven development is an increasingly popular development technique in the context of Agile methodology. It stands out because of its highly collaborative approach among the developers, QAs and users. This is a key driver in creating user centric software.

Acceptance Test driven development requires collaboration between end user team and development to help the latter with the knowledge they need to ensure the software meets the acceptance criteria.



The steps to implementing ATDD involve,

- Create tests: Acceptance tests are written in simple English and Gherkin language based on the conditions and end user cases defined by the end user/product owner.
- Run tests: The tests are run only to make them fail for obvious reasons since the required feature doesn't already exist
- Write code: Developers start writing the code knowing the criteria it will need to pass the test
- Test code: The newly written code is run through the tests in step 1 until it passes and move on the next step I.e., refactoring
- Refactoring: When all tests passes, developers can start the code refactoring exercises to meet the code quality standards.

Tools: TestNG, FitNesse, EasyB, Spectacular, Concordian, Thucydides

Behavior Driven Testing

Behavior Driven Development (BDD) is an extension of test-driven development (TDD) which emphasizes on writing the test first based on the system's behavior. It encourages a

collaborative approach from Developers, QAs and end users. Successful BDD requires clear communication and understanding of the user requirements, behaviors and acceptance criteria from the business side in an understandable language passed to the technical deliveries. This way, acceptance tests are built over time and then passes on to test teams for automation.

An example of Given-When-Then approach in simple English and Gherkin language in a BDD,

Given:
When the user has valid user credentials (Username & Password)

When:
When the user clicks on the login button

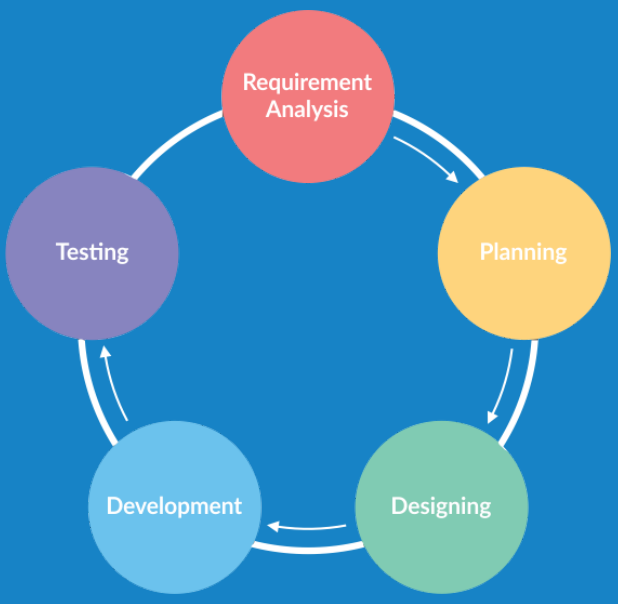
Then:
Display successful validation message

BDD testing follow the process, State behavior scenarios > write tests > Run tests and fail > write code to pass > Pass test and refactor.

Cucumber is the widely used BDD testing tool for its ability to define system/applications behavior in plain English like language (Gherkin)

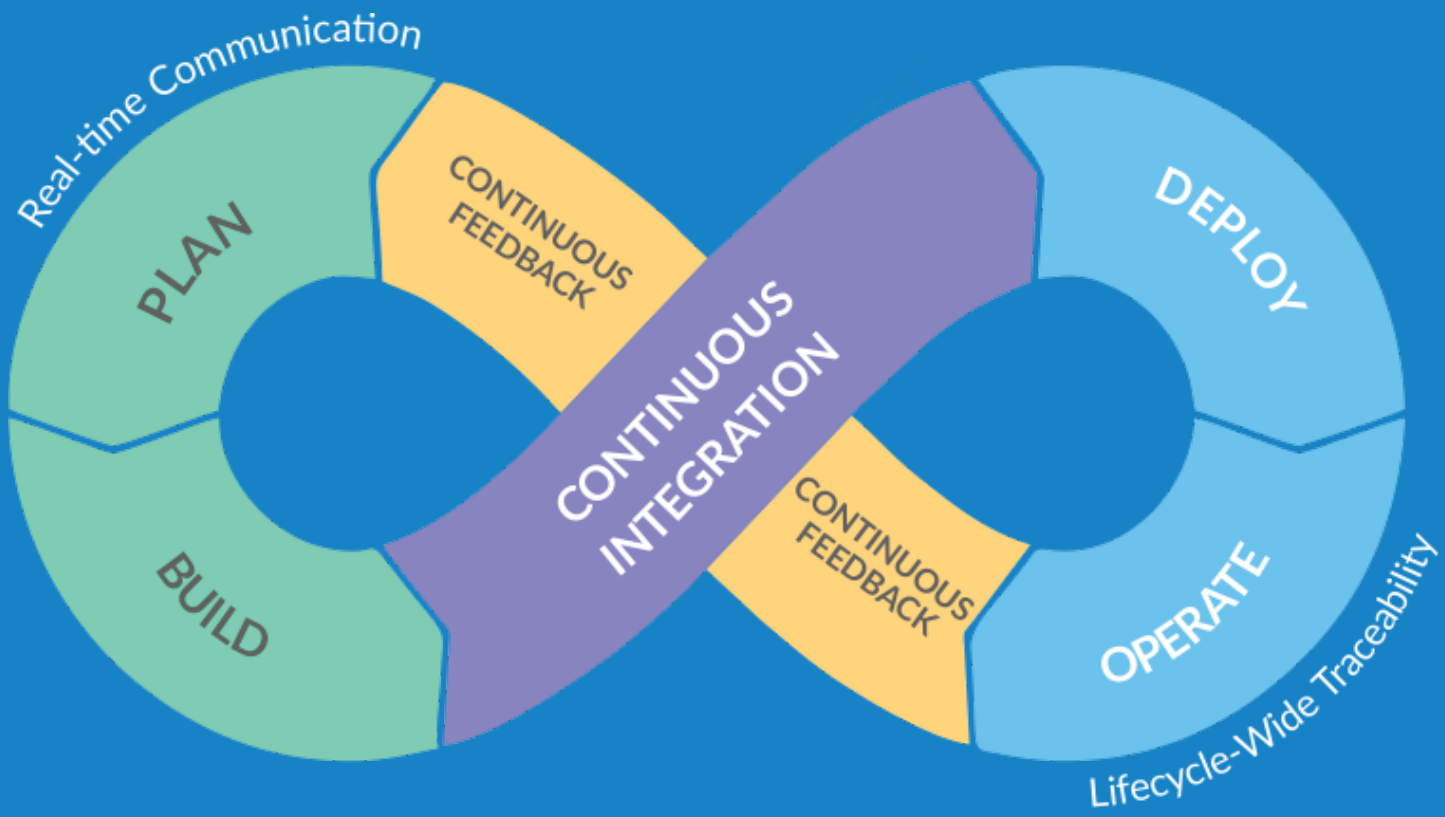
Lean/Agile Testing

Traditional waterfall teams follow the practice of separating development and testing teams in two parts: Developers first build a feature in siloes and pass it to the QA team for testing. The QA team executes detailed test plans to ensure working of the feature as intended, check regressions in existing features and also file defects.



As the product grows, the teams usually struggle to strike the right balance between releasing quality products and early time-to-market as a result of changing requirements and skip test exercises. This resulted in the agile software development.

Agile testing is a software testing process that follows the principle of agile software development. Unlike waterfall, agile brings together the development and testing team to build and ship quality products in sprints at the speed of agile. This encourages increased collaboration between developers, testers and business analysts to test the application and provide continuous feedback on the quality and fix defects in the same iteration.



DevOps Testing

DevOps is a culture, a mindset, a practice, it's a continuous everything: Continuous Integration, Continuous Testing, Continuous Delivery.

DevOps takes the agile methodology a step further by bringing the development and operations team closer. DevOps with an agile mindset is responsible for the development, testing and delivery of the software. Testing should happen at each stage in DevOps model. It starts right from continuous integration where developers will merge their code in a CI tool after which continuous (automated) testing is run followed by continuous delivery and deployment. The objective of DevOps testing is to ensure the teams take a shift-left approach and collaborate together for a continuous software delivery.

Conclusion

Software testing and methodologies have evolved a lot over the years and is no longer considered a gatekeeping activity today. The quality of the software we use every day – be it sending as simple as calling someone from your favourite app or as critical as GPS systems, the value offering lies in the quality of the testing processes it must pass before reaching its customers.

In addition to following the testing best practices and leveraging test tools, the true essence of testing can be realized only when the software testers stay in the know clearly about:

- User requirements,
- Test with user experience in mind
- Practice efficient test management and reporting
- Work closely with developers and business stakeholders
- Stay curious and updated of the changes and evolution in the testing world.

About ZUCI

Zuci is a digital organization focused on the craft of building software which we have perfected over the years. A perfect blend of design thinking, engineering perfection, and customer-centricity in our DNA has enabled us to help small, medium and large organizations with superior digital solutions. **Learn More.**

www.zucisystems.com
sales@zucisystems.com