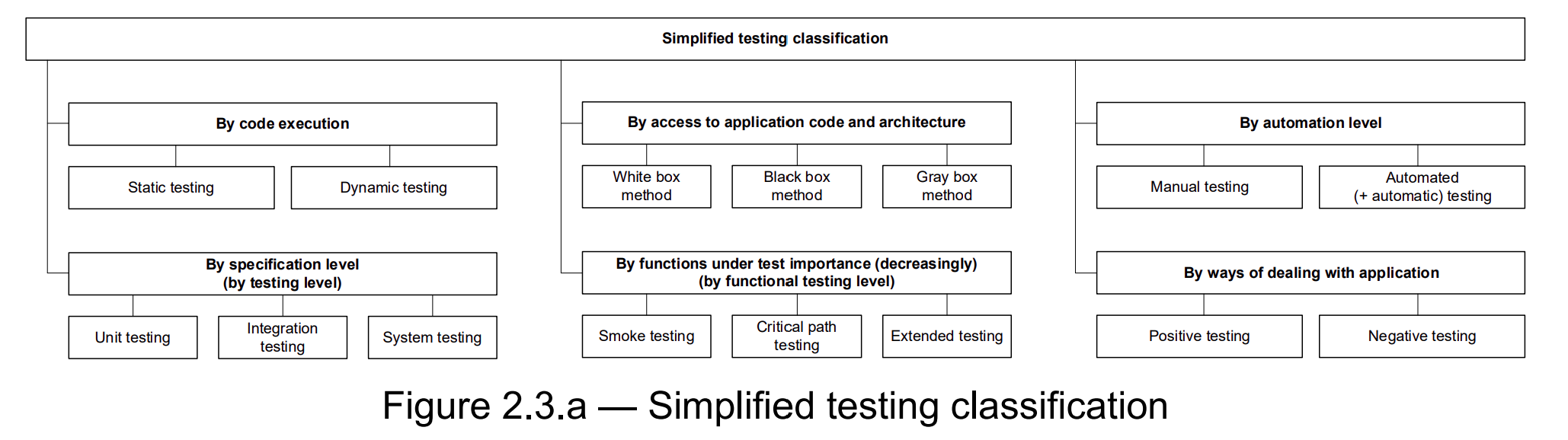
**Software testing classification**

**Simplified testing classification**

Testing can be classified in a large number of ways, and in almost every solid book on testing the author shows their own (certainly legitimate) view of the issue. The relevant material is quite extensive and complex, and a deep understanding of each item in the classification requires a certain experience, so we will divide this topic into two: now we look at the simplest, the minimum set of information necessary for the beginner tester, and in the next chapter will give a detailed classification. Use the list below as a very brief “cheat sheet to remember”. So, testing can be categorized as:

****

**By code execution:**

* Static testing — without running the code.
* Dynamic testing — with running the code.

**By access to application code and architecture:**

* White box method — there is access to the code.
* Black box method — there is no access to the code.
* Gray box method — some of the code is accessible, some of it is not.

**By automation level:**

* Manual testing — test cases are performed by a person.
* Automated testing — test cases are partially or fully performed by a special testing tool.

**By specification level (by testing level):**

* Unit testing — individual small parts of an application are tested.
* Integration testing — the interaction between several parts of the application is tested.
* System testing — the application is tested as a whole.

**By functions under test importance (decreasingly) (by functional testing level):**

* Smoke testing (be sure to study the etymology of the term — at least on Wikipedia108) — testing of the most important, most crucial functionality, the failure of which renders the very idea of using the application meaningless.
* Critical path testing — testing the functionality used by typical users in typical daily activities.
* Extended testing — testing all (remaining) functionality stated in the requirements.

**By ways of dealing with application (**Classification by aims and goals**):**

* Positive testing — all actions with the application are performed strictly according to the instructions without any unacceptable actions, incorrect data, etc. You can figuratively say that the application is tested in “greenhouse conditions”.
* Negative testing — when working with the application, operations (including incorrect ones) are performed and data are used that potentially lead to errors (a classic of the genre — division by zero).

Attention: a very frequent mistake! Negative tests do NOT imply an error in the application. On the contrary, they assume that a correctly working application will behave correctly even in a critical situation (in the example with division by zero, for example, the message “Division by zero is forbidden” is displayed).

• **Functional testing** is a testing that verifies that the application functionality works correctly Functional testing is often associated with black box testing, but white box testing can also be used to verify that functionality has been implemented correctly.

In a nutshell:

• Functional testing (as an antonym for non-functional testing) aims to verify that the functions of the application are implemented and that they work in the correct way.

• Functionality testing is aimed at the same tasks, but the focus is shifted to examining the application in its real-world environment, after localization and in similar situations.

* **Non-functional testing** is a testing of non-functional features of an application such as usability, compatibility, performance, security, etc.

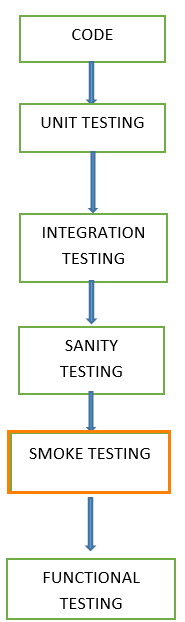
**What is Smoke Testing?**

**Smoke Testing** is a software testing process that determines whether the deployed software build is stable or not. Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as “Build Verification Testing” or “Confidence Testing.”

In simple terms, smoke tests means verifying the important features are working and there are no showstoppers in the build that is under testing. It is a mini and rapid regression test of major functionality. It is a simple test that shows the product is ready for testing. This helps determine if the build is flawed as to make any further testing a waste of time and resources.

**When do we do smoke testing**

Smoke Testing is done whenever the new functionalities of software are developed and integrated with existing build that is deployed in QA/staging environment. It ensures that all critical functionalities are working correctly or not.



In this testing method, the development team deploys the build in QA. A subsets of test cases are taken, and then testers run test cases on the build. The QA team test the application against the critical functionalities. These series of test cases are designed to expose errors that are in build. If these tests are passed, QA team continues with[Functional Testing](https://www.guru99.com/functional-testing.html).

Any failure indicates a need to handle the system back to the development team. Whenever there is a change in the build, we perform Smoke Testing to ensure the stability.

**Example**: -New registration button is added in the login window and build is deployed with the new code. We perform smoke testing on a new build.

The smoke tests qualify the build for further formal testing. The main aim of smoke testing is to detect early major issues. Smoke tests are designed to demonstrate system stability and conformance to requirements. A build includes all data files, libraries, reusable modules, engineered components that are required to implement one or more product functions.

**What happens if we don’t do Smoke testing**

If we don’t perform smoke testing in early stages, defects may be encountered in later stages where it can be costly. And the[Defect](https://www.guru99.com/defect-management-process.html)found in later stages can be show stoppers where it may affect the release of deliverables.

**Who will do Smoke Testing**

After releasing the build to QA environment, Smoke Testing is performed by QA engineers/QA lead. Whenever there is a new build, QA team determines the major functionality in the application to perform smoke testing. QA team checks for showstoppers in the application that is under testing.

Testing done in a development environment on the code to ensure the correctness of the application before releasing build to QA, this is known as Sanity testing. It is usually narrow and deep testing. It is a process which verifies that the application under development meets its basic functional requirements.

Sanity testing determines the completion of the development phase and makes a decision whether to pass or not to pass software product for further testing phase.

**Why do we do smoke testing?**

Smoke testing plays an important role in software development as it ensures the correctness of the system in initial stages. By this, we can save test effort. As a result, smoke tests bring the system to a good state. Once we complete smoke testing then only we start functional testing.

* All the show stoppers in the build will get identified by performing smoke testing.
* Smoke testing is done after the build is released to QA. With the help of smoke testing, most of the defects are identified at initial stages of software development.
* With smoke testing, we simplify the detection and correction of major defects.
* By smoke testing, QA team can find defects to the application functionality that may have surfaced by the new code.
* Smoke testing finds the major severity defects.

**Example 1:**Logging window: Able to move to next window with valid username and password on clicking submit button.

**Example 2:**Userunable to sign out from the webpage.

**How to do Smoke Testing?**

Smoke Testing is usually done manually though there is a possibility of accomplishing the same through automation. It may vary from organization to organization.

**Manual Smoke testing**

In general, smoke testing is done manually. It approaches varies from one organization to other. Smoke testing is carried to ensure the navigation of critical paths is as expected and doesn’t hamper the functionality. Once the build is released to QA, high priority functionality test cases are to be taken and are tested to find the critical defects in the system. If the test passes, we continue the functional testing. If the test fails, the build is rejected and sent back to the development team for correction. QA again starts smoke testing with a new build version. Smoke testing is performed on new build and will get integrated with old builds to maintain the correctness of the system. Before performing smoke testing, QA team should check for correct build versions.

**Smoke testing by Automation**

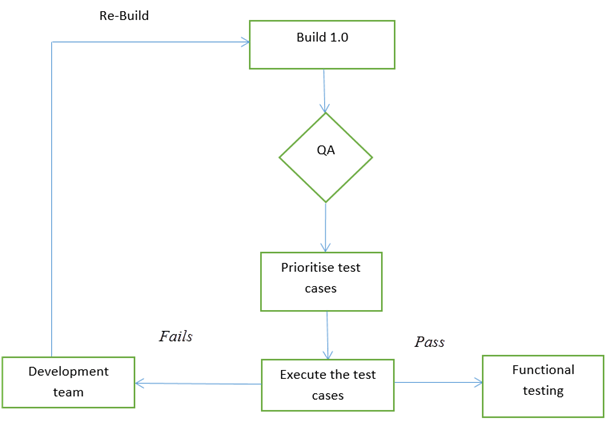
[Automation Testing](https://www.guru99.com/automation-testing.html) is used for [Regression Testing](https://www.guru99.com/regression-testing.html). However, we can also use a set of automated test cases to run against Smoke Test. With the help of automation tests, developers can check build immediately, whenever there is a new build ready for deployment.

Instead of having repeated test manually whenever the new software build is deployed, recorded smoke test cases are executed against the build. It verifies whether the major functionalities still operates properly. If the test fails, then they can correct the build and redeploy the build immediately. By this, we can save time and ensure a quality build to the QA environment.

Using an automated tool, test engineer records all manual steps that are performed in the software build.

**Smoke testing cycle**

Below flow chart shows how Smoke Testing is executed. Once the build is deployed in QA and, smoke tests are passed we proceed for functional testing. If the smoke test fails, we exit testing until the issue in the build is fixed.



Smoke test cycle

**Advantages of Smoke testing**

Here are few advantages listed for Smoke Testing.

* Easy to perform testing
* Defects will be identified in early stages.
* Improves the quality of the system
* Reduces the risk
* Progress is easier to access.
* Saves test effort and time
* Easy to detect critical errors and correction of errors.
* It runs quickly
* Minimises integration risks

**Sample Smoke Test Cases Example**

## Sample Smoke Test Cases Example

| **T.ID** | **TEST SCENARIOS** | **DESCRIPTION** | **TEST STEP** | **EXPECTED RESULT** | **ACTUAL RESULT** | **STATUS** |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | Valid login credentials | Test the login functionality of the web application to ensure that a registered user is allowed to login with username and password | 1.Launch the application 2.Navigate the login page 3.Enter valid username 4.Enter valid password 5.Click on login button | Login should be success | as expected | Pass |
| 2 | Adding item functionality | Able to add item to the cart | 1.Select categories list 2.Add the item to cart | Item should get added to the cart | Item is not getting added to the cart | Fail |
| 3 | Sign out functionality | Check sign out functionality | 1. select sign out button | The user should be able to sign out. | User is not able to sign out | Fail |

**Conclusion**

In Software Engineering, Smoke testing should be performed on each and every build without fail as it helps to find defects in early stages. Smoke test activity is the final step before the software build enters the system stage. Smoke tests must be performed on each build that is turned to testing. This applies to new development and major and minor releases of the system.

Before performing smoke testing, QA team must ensure the correct build version of the application under test. It is a simple process which takes a minimum time to test the stability of the application.

Smoke tests can minimise test effort, and can improve the quality of the application. Smoke testing can be done either manually or by automation depending on the client and the organization.

**What is Regression Testing?**

**Regression Testing** is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features. Regression Testing is nothing but a full or partial selection of already executed test cases that are re-executed to ensure existing functionalities work fine.

This testing is done to ensure that new code changes do not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

**Why Regression Testing?**

There is a **need for regression testing** whenever the code is changed, and you need to determine whether the modified code will affect other parts of the software application. Moreover, regression testing is needed when a new feature is added to the software application. Regression tests may also be performed when a functional or performance defect/issue is fixed.

**How to do Regression Testing**

In order **to do Regression testing** process, we need to first debug the code to identify the bugs. Once the bugs are identified, required changes are made to fix it, then the regression testing is done by selecting relevant test cases from the test suite that covers both modified and affected parts of the code.

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of existing features. These modifications may cause the system to work incorrectly. Therefore, Regression Testing becomes necessary. Regression Testing can be carried out using the following techniques:



**Retest All**

This is one of the methods for Regression Testing in which all the tests in the existing test bucket or suite should be re-executed. This is very expensive as it requires huge time and resources.

**Regression Test Selection**

Regression Test Selection is a technique in which some selected test cases from test suite are executed to test whether the modified code affects the software application or not. Test cases are categorized into two parts, reusable test cases which can be used in further regression cycles and obsolete test cases which cannot be used in succeeding cycles.

**Prioritization of Test Cases**

Prioritize the test cases depending on business impact, critical & frequently used functionalities. Selection of test cases based on priority will greatly reduce the regression test suite.

**Selecting test cases for regression testing**

It was found from industry data that a good number of the defects reported by customers were due to last minute bug fixes creating side effects and hence selecting the[Test Case](https://www.guru99.com/test-case.html)for regression testing is an art and not that easy.  Effective Regression Tests can be done by selecting the following test cases –

* Test cases which have frequent defects
* Functionalities which are more visible to the users
* Test cases which verify core features of the product
* Test cases of Functionalities which has undergone more and recent changes
* All Integration Test Cases
* All Complex Test Cases
* Boundary value test cases
* A sample of Successful test cases
* A sample of Failure test cases

**Regression Testing Tools**

If your software undergoes frequent changes, regression testing costs will escalate. In such cases, Manual execution of test cases increases test execution time as well as costs. Automation of regression test cases is the smart choice in such cases.  The extent of automation depends on the number of test cases that remain re-usable for successive regression cycles.

Following are the most important tools used for both functional and regression testing in software engineering:

**1) [testRigor](https://bit.ly/32ye79O" \t "_blank)**

[testRigor](https://bit.ly/32ye79O) helps you to directly express tests as executable specifications in plain English. Users of all technical abilities are able to build end-to-end tests of any complexity covering mobile, web, and API steps in one test. Test steps are expressed on the end-user level instead of relying on details of implementation like XPaths or CSS Selectors.

[](https://bit.ly/32ye79O)

**Features:**

* Free forever public version
* Test cases are in English
* Unlimited users & Unlimited tests
* The easiest way to learn automation
* Recorder for web steps
* Integrations with CI/CD and Test case management
* Email & SMS testing
* Web + Mobile + API steps in one test

**2) [Avo Assure](https://guru99.link/recommends-avoautomation-regression-testing" \t "_blank)**

[Avo Assure](https://guru99.link/recommends-avoautomation-regression-testing) is a technology agnostic, no-code test automation solution that helps you test end-to-end business processes with a few clicks of the buttons. This makes regression testing more straightforward and faster.

[](https://guru99.link/recommends-avoautomation-regression-testing)

**Features**

* Auto generate test cases with a 100% no-code approach
* Test across the web, desktop, mobile, ERP applications, Mainframes, associated emulators, and more with a single solution.
* Enable accessibility testing
* Execute test cases in a single VM independently or in parallel with Smart Scheduling
* Integrate with Jira, Jenkins, ALM, QTest, Salesforce, Sauce Labs, TFS, etc.
* Define test plans and design test cases through the Mind maps feature

**3)**[**Subject7**](https://guru99.live/wZlz6b)

[Subject7](https://guru99.live/wZlz6b) is a cloud-based, “true codeless” test automation solution that unifies all testing in a single platform and empowers anyone to become an automation expert. Our easy-to-use software enables fast, easy, and sophisticated authoring of regression tests without writing a line of code, and high scale execution that runs thousands of nightly tests.

[](https://guru99.live/wZlz6b)

**Features:**

* Integrates easily with DevOps/Agile tooling using native plugins, in-app integrations, and open APIs.
* High-scale parallel execution in the cloud or on-prem with enterprise-grade security.
* Flexible reporting of defects, with video capture of results.
* Simple, non-metered pricing, delivering financial predictability.
* SOC2 Type2 compliant

[**Selenium**](https://www.guru99.com/selenium-tutorial.html): This is an open source tool used for automating web applications. Selenium can be used for browser-based regression testing.

[**Quick Test Professional (QTP)**](https://www.guru99.com/quick-test-professional-qtp-tutorial.html): HP Quick Test Professional is automated software designed to automate functional and regression test cases. It uses[VBScript](https://www.guru99.com/vbscript-tutorials-for-beginners.html)language for automation. It is a Data-driven, Keyword based tool.

**Rational Functional Tester (RFT)**: IBM’s rational functional tester is a[Java](https://www.guru99.com/java-tutorial.html)tool used to automate the test cases of software applications. This is primarily used for automating regression test cases and it also integrates with Rational Test Manager.

**Regression Testing and Configuration Management**

Configuration Management during Regression Testing becomes imperative in Agile Environments where a code is being continuously modified. To ensure effective regression tests, observe the following :

* Code being regression tested should be under a configuration management tool
* No changes must be allowed to code, during the regression test phase.  Regression test code must be kept immune to developer changes.
* The database used for regression testing must be isolated. No database changes must be allowed

**Difference between Re-Testing and Regression Testing**

Retesting means testing the functionality or bug again to ensure the code is fixed. If it is not fixed,[Defect](https://www.guru99.com/defect-management-process.html)needs to be re-opened. If fixed, Defect is closed.

Regression testing means testing your software application when it undergoes a code change to ensure that the new code has not affected other parts of the software.

Also, Check out the complete list of differences over [here](https://www.guru99.com/re-testing-vs-regression-testing.html).

**Challenges in Regression Testing**

Following are the major testing problems for doing regression testing:

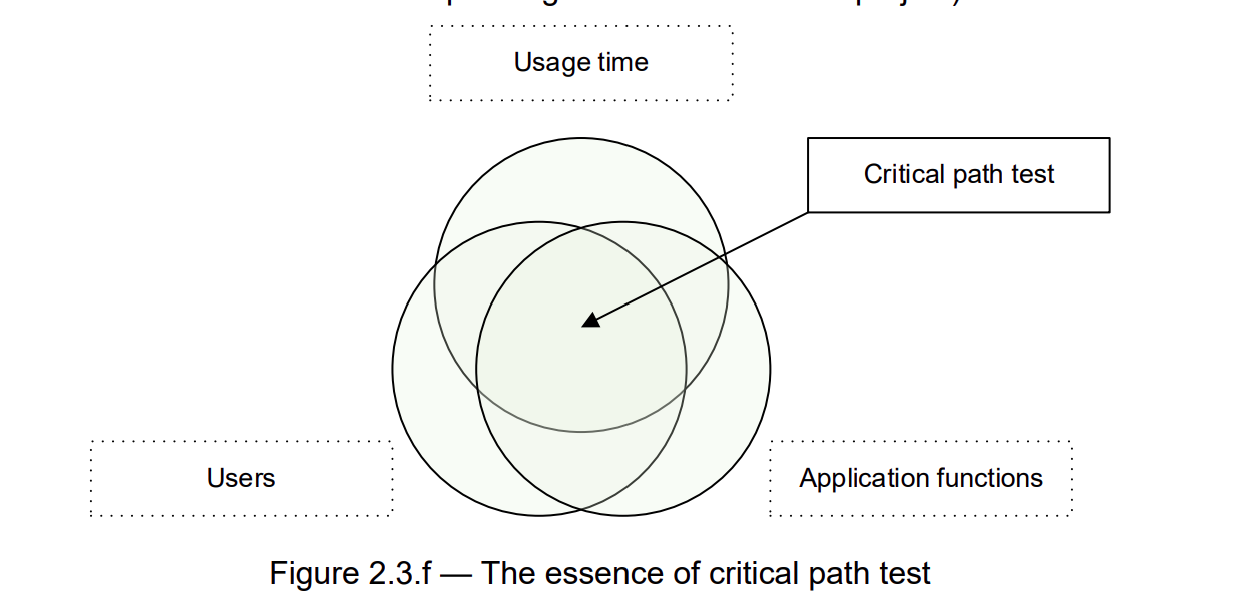
* With successive regression runs, test suites become fairly large.  Due to time and budget constraints, the entire regression test suite cannot be executed
* Minimizing the test suite while achieving maximum[Test coverage](https://www.guru99.com/test-coverage-in-software-testing.html)remains a challenge
* Determination of frequency of Regression Tests, i.e., after every modification or every build update or after a bunch of bug fixes, is a challenge.

**Conclusion**

* Regression Testing Meaning – Regression testing is a type of software testing that ensures an application still functions as expected after improvements, any code changes, or updates.
* An effective regression strategy, save organizations both time and money. As per one of the case study in banking domain, regression saves up to 60% time in bug fixes(which would have been caught by regression tests) and 40%  in money

**Critical path test**

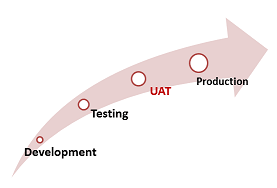
Critical path test is aimed at examining the functionality used by typical users in a typical day-to-day activity. As can be seen from the definition in the footnote142 (142 Critical path. Longest sequence of activities in a project plan which must be completed on time for the project to complete on due date. An activity on the critical path cannot be started until its predecessor activity is complete; if it is delayed for a day, the entire project will be delayed for a day unless the activity following the delayed activity is completed a day earlier. [https://everhour.com/blog/how-to-calculate), the idea itself is borrowed from project management and transformed in the context of testing to the following: there are most users who most often use some subset of application functions (see figure 2.3.f). Those are exactly the functions that need to be tested, once we have ensured that the application “generally works” (the smoke test was successful). If for some reason the application does not perform these functions, or performs them incorrectly, a great many users will not be able to achieve many of their goals. The threshold for a successful “critical path test” metric is already slightly lower than for the smoke test, but still quite high (typically around 70–80–90 % — depending on the nature of the project).



**What is UAT?**

**User Acceptance Testing (UAT)** is a type of testing performed by the end user or the client to verify/accept the software system before moving the software application to the production environment. UAT is done in the final phase of testing after functional, integration and system testing is done.

**Purpose of UAT**



The main **Purpose of UAT** is to validate end to end business flow. It does not focus on cosmetic errors, spelling mistakes or system testing. User Acceptance Testing is carried out in a separate testing environment with production-like data setup. It is kind of black box testing where two or more end-users will be involved.

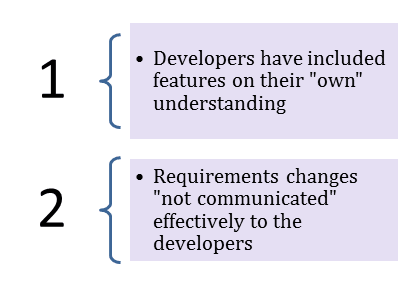
UAT is performed by –

 Client

 End users

**Need of User Acceptance Testing**

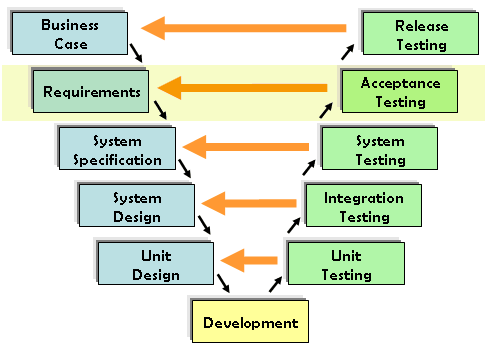
**Need of User Acceptance Testing** arises once software has undergone Unit, Integration and System testing because developers might have built software based on requirements document by their own understanding and further required changes during development may not be effectively communicated to them, so for testing whether the final product is accepted by client/end-user, user acceptance testing is needed.



* Developers code software based on requirements document which is their “own” understanding of the requirements and **may not actually be what the client needs from the software**.
* Requirements changes during the course of the project may not be communicated effectively to the developers.

**Acceptance Testing and V-Model**

In VModel, User acceptance testing corresponds to the requirement phase of the Software Development life cycle(SDLC).



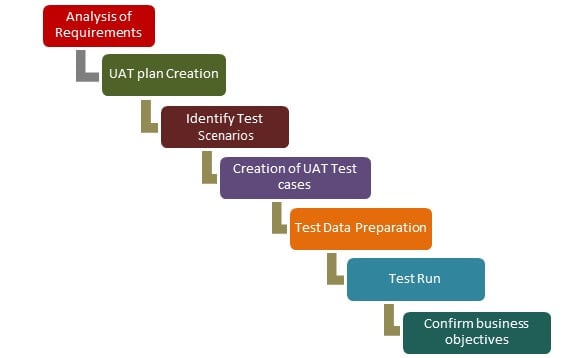
**Prerequisites of User Acceptance Testing:**

Following are the entry criteria for User Acceptance Testing:

* Business Requirements must be available.
* Application Code should be fully developed
* Unit Testing, Integration Testing & System Testing should be completed
* No Showstoppers, High, Medium defects in System Integration Test Phase –
* Only Cosmetic error is acceptable before UAT
* Regression Testing should be completed with no major defects
* All the reported defects should be fixed and tested before UAT
* Traceability matrix for all testing should be completed
* UAT Environment must be ready
* Sign off mail or communication from System Testing Team that the system is ready for UAT execution

**How to execute UAT Tests**

UAT is done by the intended users of the system or software. This type of Software Testing usually happens at the client location which is known as Beta Testing. Once Entry criteria for UAT are satisfied, following are the tasks need to be performed by the testers:

UAT Process

* Analysis of Business Requirements
* Creation of UAT test plan
* Identify Test Scenarios
* Create UAT Test Cases
* Preparation of Test Data(Production like Data)
* Run the Test cases
* Record the Results
* Confirm business objectives

**Step 1) Analysis of Business Requirements**

One of the most important activities in the UAT is to identify and develop test scenarios. These test scenarios are derived from the following documents:

* Project Charter
* Business Use Cases
* Process Flow Diagrams
* Business Requirements Document(BRD)
* System Requirements Specification(SRS)

**Step 2) Creation of UAT Plan:**

The UAT test plan outlines the strategy that will be used to verify and ensure an application meets its business requirements. It documents entry and **exit criteria for UAT, Test scenarios and test cases approach and timelines of testing**.

**Step 3) Identify Test Scenarios and Test Cases:**

Identify the test scenarios with respect to high-level business process and create test cases with clear test steps. Test Cases should sufficiently cover most of the UAT scenarios. Business Use cases are input for creating the test cases.

**Step 4) Preparation of Test Data:**

It is best advised to use live data for UAT. Data should be scrambled for privacy and [security](https://www.guru99.com/ethical-hacking-tutorials.html) reasons. Tester should be familiar with the database flow.

**Step 5) Run and record the results:**

Execute test cases and report bugs if any. Re-test bugs once fixed. [Test Management](https://www.guru99.com/test-management.html) tools can be used for execution.

**Step 6) Confirm Business Objectives met:**

Business Analysts or UAT Testers needs to send a sign off mail after the UAT testing. After sign-off, the product is good to go for production. Deliverables for UAT testing are Test Plan, UAT Scenarios and Test Cases, Test Results and Defect Log

**Exit criteria for UAT:**

Before moving into production, following needs to be considered:

* No critical defects open
* Business process works satisfactorily
* UAT Sign off meeting with all stakeholders

**Qualities of UAT Testers:**



UAT Tester should possess good knowledge of the business. He should be independent and think as an **unknown user to the system**. Tester should be Analytical and Lateral thinker and combine all sort of data to make the UAT successful.

Tester or Business Analyst or Subject Matter Experts who understand the business requirements or flows can prepare test and data which are realistic to the business.

**Best Practices:**

Following points needs to be considered to make UAT Success:

* Prepare UAT plan early in the project life cycle
* Prepare Checklist before the UAT starts
* Conduct Pre-UAT session during System Testing phase itself
* Set the expectation and define the scope of UAT clearly
* Test End to End business flow and avoid system tests
* Test the system or application with real-world scenarios and data
* Think as an Unknown user to the system
* Perform Usability Testing
* Conduct Feedback session and meeting before moving to production

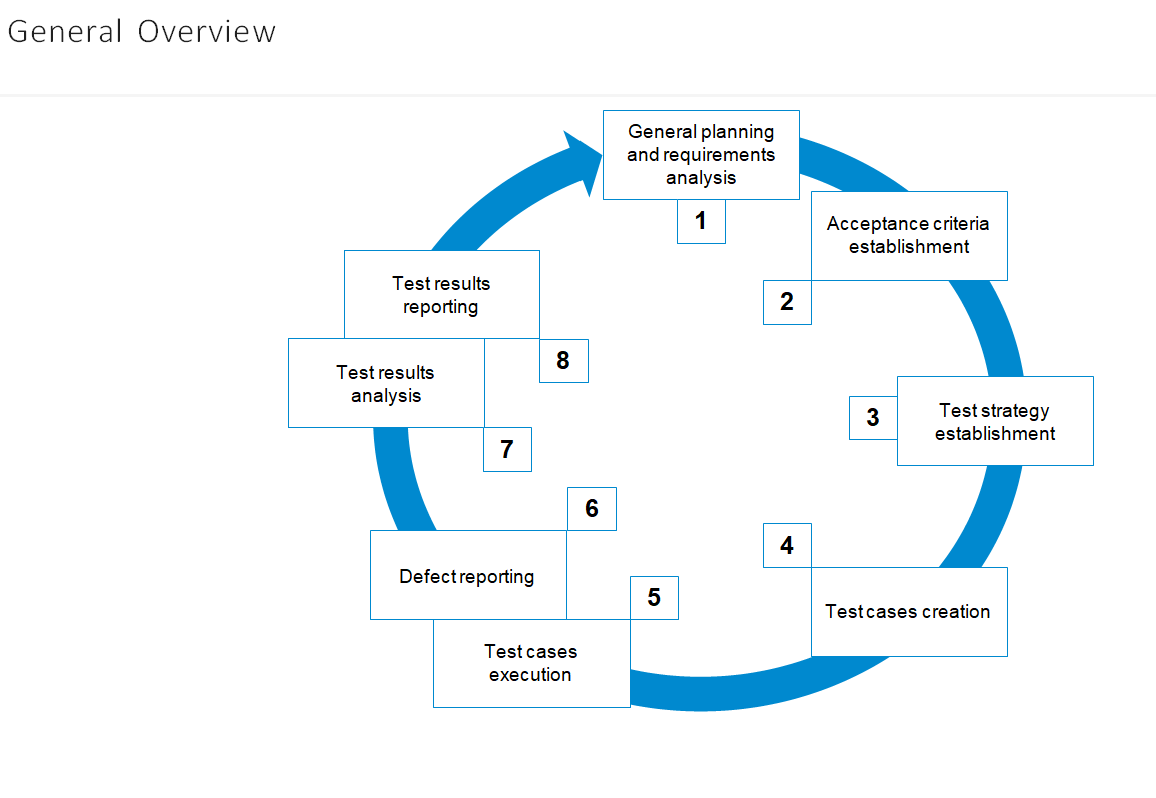
**Conclusion:**

* In Software Engineering, Full form of UAT is User Acceptance Testing.
* UAT is one of the many flavours of testing that has emerged over last twenty-five years.
* With UAT, the client can be sure “What to expect” from the product rather than assuming.
* The benefit of UAT is that there will be no surprises when the product is released to the market.

**Basic Software Testing Terminology**

* **Testing** – the process concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demonstrate that they are fit for purpose and to detect defects.
* **Quality** – the degree to which a component, system or process meets specified requirements and/or user/customer needs and expectations.
* **Quality management** – Coordinated activities to direct and control an organization with regard to quality that include quality control and quality assurance improvement.
* **Quality assurance** – part of quality management focused on providing confidence that quality requirements will be fulfilled.
* **Quality control** – A set of activities designed to evaluate the quality of a component or system.
* **Defect** – an imperfection or deficiency in a work product where it does not meet its requirements or specifications.
* **Expected result** – the predicted observable behavior of a component or system executing under specified conditions, based on its specification or another source.
* **Actual result** – the behaviour produced/observed when a component or system is tested.
* **Checklist** – a set of ideas.
* **Test case** – a set of preconditions, inputs, actions (where applicable), expected results and post conditions, developed based on test conditions.
* **Test suite** – a set of test cases or test procedures to be executed in a specific test cycle.

**Software Testing Lifecycle**



**1. General planning and requirements analysis**

Here we have to find out: what to test; how much work is ahead; what difficulties we may face; do we have all necessary resources; are requirements good enough.

**2. Acceptance criteria establishment**

Here we have to establish or adjust metrics and criteria for test process to start, pause, resume, complete or abort. We also have to know key quality criteria and goals for the current test cycle.

**3. Test strategy establishment**

Once again we return to planning to find out HOW shall we achieve all those goals and criteria from the previous step. Here we speak about approaches, tools, schedule, roles, responsibility and so on.

**4. Test cases creation**

Here we create, review, adjust, rework (and so on) checklists, test cases, test scenarios and other similar artifacts.

1. **Test cases execution**
2. **Defect reporting**

These two stages are inseparable as we report defects during test cases execution and defects detection.

