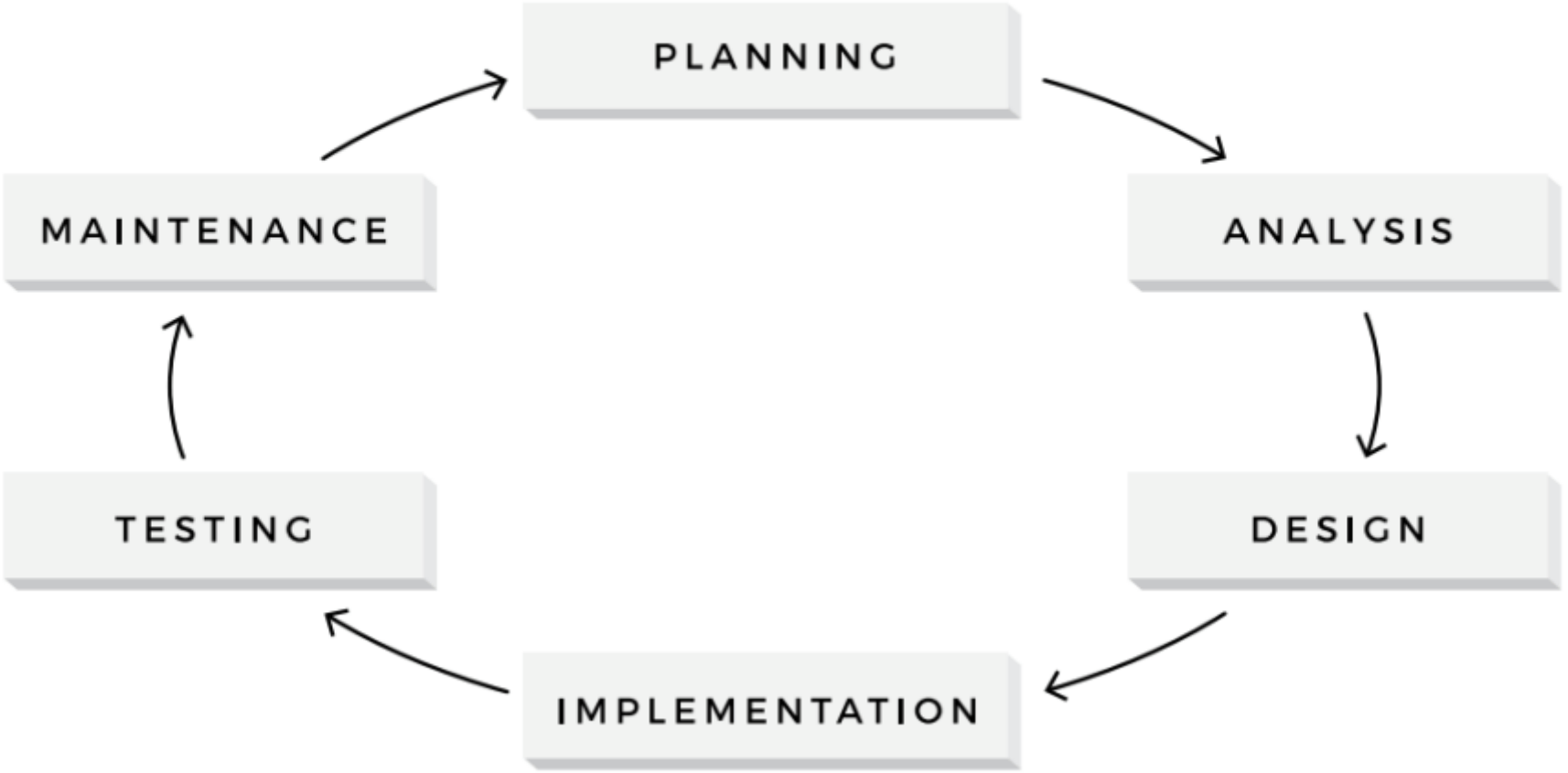
Testing Methods

Introduction to STLC and V Model



Importance of Test Process:

- It streamlines the work that is being performed by the team members.
- It brings consistency across the team as well as the way they work.
- Makes future maintenance of the product easier.
- Following a fundamental testing process results in better product.
- It helps plan and manage the testing activities.
- Well-planned and systematic process enables the team to detect defects more rapidly.

Software Testing Life Cycle (STLC) is defined as a sequence of activities conducted to perform Software Testing

Each of these stages has a definite Entry and Exit criteria, Activities & Deliverables associated with it for all levels in the Software Testing Life Cycle (STLC)

- **Entry Criteria:** Entry Criteria gives the prerequisite items that must be completed before testing can begin.
- **Exit Criteria:** Exit Criteria defines the items that must be completed before testing can be concluded.

Requirement Analysis

- During this phase, test team studies the requirements from a testing point of view to identify the testable requirements.
- The QA team may interact with various stakeholders (Client, Business Analyst, Technical Leads, System Architects etc) to understand the requirements in detail.
- Requirements could be either Functional or Non Functional.

Activities

- Identify types of tests to be performed.
- Gather details about testing priorities and focus.
- Prepare Requirement Traceability Matrix (RTM).
- Identify test environment details where testing is supposed to be carried out.
- Automation feasibility analysis (if required).

Deliverables

- **RTM**
- Automation feasibility report. (if applicable)

Test Planning

- Determine effort and cost estimates for the project and preparation and finalization of the Test Plan is done.
- In this phase, Test Strategy is also determined.

Activities

- Preparation of test plan/strategy document for various types of testing
- Test tool selection
- Test effort estimation
- Resource planning and determining roles and responsibilities.
- Training requirement

Deliverables

- Test plan /strategy document.
- Effort estimation document.

Test Case Development

- This phase involves the creation, verification and rework of test cases & test scripts.
- Test data, is identified/created and is reviewed and then reworked as well.

Activities

- Create test cases, automation scripts (if applicable)
- Review and baseline test cases and scripts
- Create test data (If Test Environment is available)

Deliverables

- Test cases/scripts
- Test data

Test Environment Setup

- The software and hardware conditions under which a work product is tested.
- Test environment set-up ***can be done in parallel with Test Case Development Stage***.
- ***Test team may not be involved in this activity*** if the customer/development team provides the test environment in which case the test team is required to do a readiness check (smoke testing) of the given environment.

Activities

- Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.
- Setup test Environment and test data
- Perform smoke test on the build

Deliverables

- Environment ready with test data set up
- Smoke Test Results.

Test Execution

- The testers will carry out the testing based on the test plans and the test cases prepared.
- Bugs will be reported back to the development team for correction and retesting will be performed.

Activities

- Execute tests as per plan
- Document test results, and log defects for failed cases
- Map defects to test cases in RTM
- Retest the Defect fixes
- Track the defects to closure

Deliverables

- Completed RTM with the execution status
- Test cases updated with results
- Defect reports

Test Cycle Closure

- Analyse testing artefacts to identify strategies that have to be implemented in the future.
- Remove the process bottlenecks for future test cycles and share best practices for any similar projects in the future.

Activities

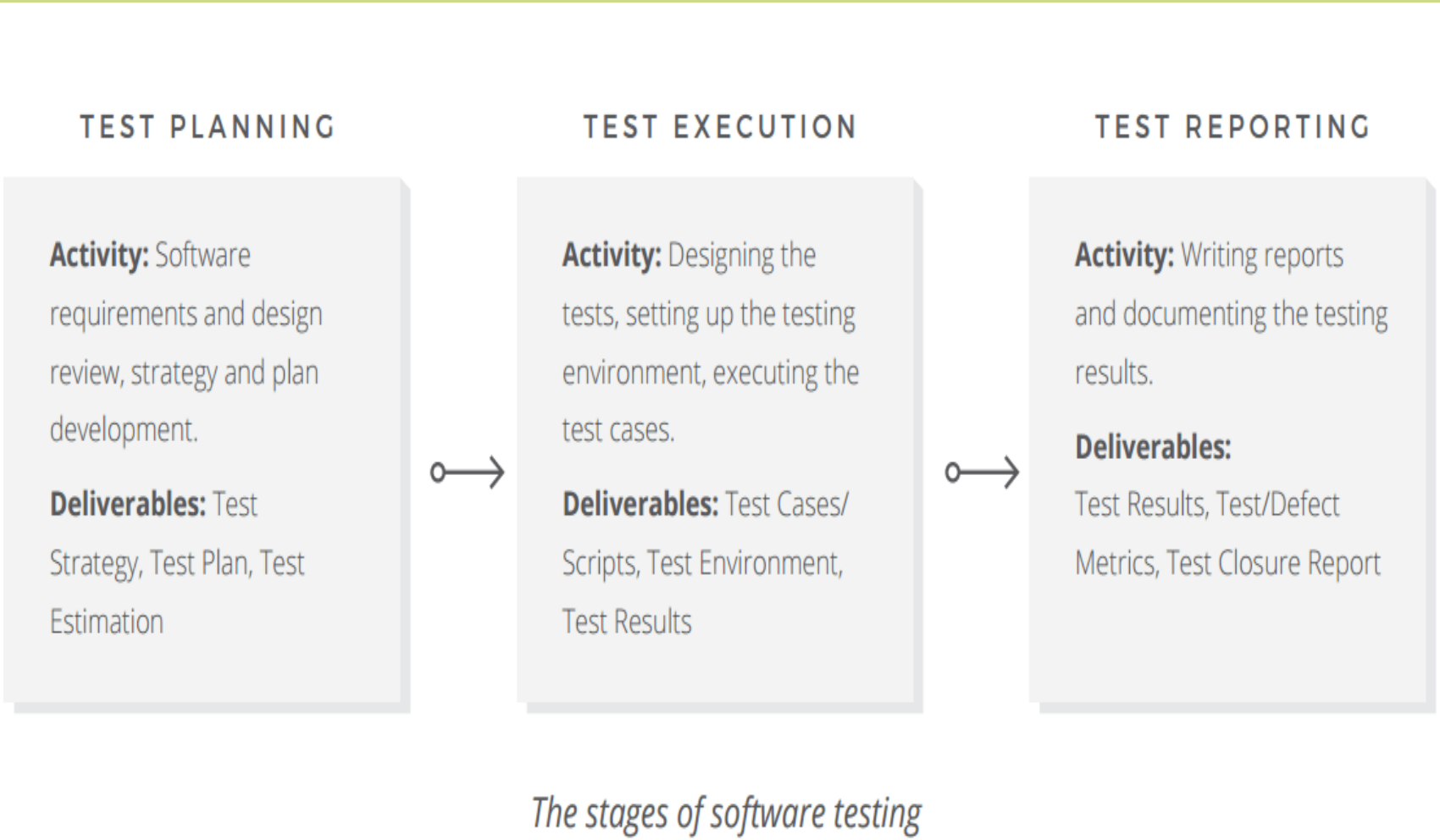
- Evaluate cycle completion criteria based on Time, Test coverage, Cost Software, Critical Business Objectives, Quality
- Prepare test metrics based on the above parameters.
- Document the learning out of the project & Prepare Test closure report.
- Qualitative and quantitative reporting of quality of the work product to the customer.
- Test result analysis to find out the defect distribution by type and severity.

•Deliverables

•Test Closure report

•Test metrics

The Process of Software Testing in Practice



The stages of software testing

As any other formal process, testing activities are typically preceded by thorough preparations and planning. The main goal of this stage is to make sure the team understands the customer objectives, the main purpose of the product, the possible risks they need to address, and the outcomes they expect to achieve. One of the documents created at this stage, **the mission or assignment of testing**, serves to solve this task. It helps align the testing activities with the overall purpose of the product and coordinates the testing effort with the rest of the team's work.

Test strategy is another artefact of the planning stage. James Bach, a testing guru who created the Rapid Software Testing course, identifies the purpose of a test strategy as "*to clarify the major tasks and challenges of the test project*." Depending on when exactly in the process they are used, the strategies can be classified as preventive or reactive.

Strategy	Character	Primary Focus Area	Use Case
Analytical	Preventive	The strategy focuses on risk and requirements analysis to create a basis for planning, building and estimating the tests.	When building a well-defined product from scratch.
Model-Based	Preventive	The system is tested in accordance with the predefined model and should completely correspond to it in order to be considered valid.	When using an existing product as a basis for a new one, or enhancing the legacy system.
Methodical	Preventive	This strategy adheres to a custom pre-planned, systematic approach.	Often used in heavily-regulated industries, to build a product that complies with the requirements.
Process/ Standard-Compliant	Preventive	Unlike the previous one, this strategy relies on a standard strategy, with little or no adaptation.	Used by the teams that lack experience or time for building a custom testing approach.
Dynamic	Reactive	It prioritizes finding as many errors and defects as possible (the attack-based approach and the exploratory approach).	When there is a need to find and fix the issues with minimum time and effort.
Consultative/ Directed	Reactive	It relies on the users or developers to define the areas of testing or even to handle the tests themselves.	Applied to domain-specific products, that require additional expert guidance.
Regression-averse	Reactive	This strategy prioritizes the automation of functional tests either before the release or after.	Best used with live, well-established products.

While a test strategy is a high-level document, **TEST PLAN** has a more hands-on approach, describing in detail what to test, how to test, when to test and who will do the test.

According to the IEEE standard for software test documentation, a **test plan** document should contain the following information:

- Test plan identifier
- Introduction
- References (list of related documents)
- Test items (the product and its versions)
- Features to be tested
- Features not to be tested
- Item pass or fail criteria
- Test approach (testing levels, types, techniques)
- Suspension criteria
- Deliverables (Test Plan (this document itself), Test Cases, Test Scripts, Defect/Enhancement Logs, Test Reports)
- Test environment (hardware, software, tools)
- Estimates
- Schedule
- Staffing and training needs
- Responsibilities
- Risks
- Assumptions and Dependencies
- Approvals

Design and Execution

- define what is subject to testing
- QA teams develop test cases.
- a test case describes the preconditions, desired outcomes, and postconditions of a specific test scenario

Documentation and Reporting

- The testing logs and status reports are documented.
- Every issue found in the product should be reported and handled accordingly.
- The test summary and test closure reports are prepared and provided to the stakeholders.
- Define and document the issues that occurred during the development and improve the process.

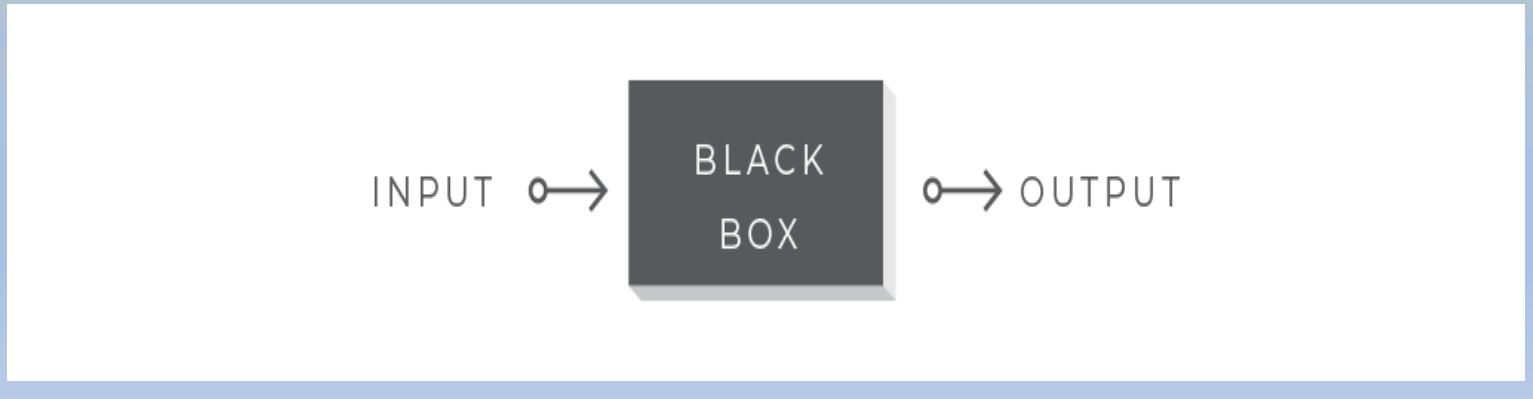
The software testing process identifies two broad categories:

- **StaticTesting:** examines the source code and software project documents to catch and prevent defects early in the software testing life cycle.
- **DynamicTesting:** where software is tested during execution.These can be described by methods, levels, and types of underlying QA activities

Software testing methods are the ways the tests are conducted.

Black Box Testing

This method gets its name because a QA engineer focuses on the inputs and the expected outputs without knowing how the application works internally and how these inputs are processed. The purpose of this method is to check the functionality of the software making sure that it works correctly and meets user demands. This method can be applied to any testing level but is used mostly for system and user acceptance testing.



Types of Black Box Testing:

- Functional Testing
- Non-functional Testing

Functional testing: what the system actually does is functional testing. To verify that each function of the software application behaves as specified in the requirement document. Testing all the functionalities by providing appropriate input to verify whether the actual output is matching the expected output or not. The testers need not concern about the source code of the application.

Non-functional testing: how well the system performs is non-functionality testing. Non-functional testing refers to various aspects of the software such as performance, load, stress, scalability, security, compatibility etc., Main focus is to improve the user experience on how fast the system responds to a request.

White Box Testing

Unlike black box testing, this method requires profound knowledge of the code as it entails testing of some structural part of the application. Therefore, generally, the developers directly involved in writing code are responsible for this type of testing. The purpose of white box testing is to enhance security, the flow of inputs/outputs through the application, and to improve design and usability. This method is mainly used at the unit and integration testing levels.

Grey Box Testing

This method is a combination of the previous two, since it involves testing of both functional and structural parts of the application. Using this method, an experienced tester has partial knowledge of the internal application structure and based on this knowledge can design test cases while still testing from the black-box perspective. This method is mostly applicable to the integration testing level.

Ad Hoc Testing

This is an informal testing method as it's performed without planning and documentation. Conducting tests informally and randomly without any formal, expected results, the tester improvises the steps and arbitrarily executes them. Though defects found with this method are more difficult to reproduce given the absence of written test cases, this approach helps find important defects quickly, something which cannot be done with formal methods.

Software testing levels describe stages of software development when testing is conducted. there are four progressive testing levels based on the area they focus *unit testing*, *integration testing*, *system testing*, and *user acceptance testing (UAT)*.

Component/Unit Testing

The smallest testable part of the software system is often referred to as a unit. Therefore, this testing level is aimed at examining every single unit of a software system in order to make sure that it meets the original requirements and functions as expected. Unit testing is commonly performed early in the development process by the engineers themselves, not the testing team.

Integration Testing

Verify whether the combined units work well together as a group. It is aimed at detecting the flaws in the interactions between the units within a module.

Two main approaches to this testing:

- **The bottom-up integration testing** starts with unit tests, successively increasing the complexity of the software modules under test.
- **The top-down integration testing** takes the opposite approach, focusing on high-level combinations first and examining the simple ones later.

System Testing

A complete software system is tested as a whole. This stage verifies the product's compliance with the functional and technical requirements and overall quality standards. System testing should be performed as close to the real business use scenario as possible.

User Acceptance Testing

This is the last stage of the testing process, where the product is validated against the end user requirements and for accuracy. This testing level focuses on overall system quality, from content and UI to performance issues. The acceptance stage might be followed by an alpha and beta testing, allowing a small number of actual users to try out the software before it is officially released.

Software testing types are the approaches and techniques that are applied at a given level using an appropriate method to address the test requirements in the most efficient manner. They are vast in number while serving different objectives.

Testing type	Object	Method used	Levels of testing
Functional Testing	Testing software functions	Black Box	User acceptance System
Performance Testing	Testing responsiveness and stability of the system performance under a certain load	Black Box	Any level
Use case Testing	Checking that the path used by the user is working as intended	Black Box	User acceptance System Integration
Exploratory Testing	Validating user experience	Ad hoc	User acceptance System
Usability testing	Checking that the system is easy to use	Black Box	User acceptance System
Security testing	Protecting the system	White Box	System

Typically, the process of functional testing comprises the following set of actions:

1. Outlines the functions for the software to perform
2. Composes the input data depending on function specifications
3. Determines the output depending on function specifications
4. Executes the test case
5. Juxtaposes the received and expected outputs

Performance Testing is aimed at investigating the responsiveness and stability of the system performance under a certain load.

Depending on the workload, a system behaviour is evaluated by different kinds of performance testing:

- Load Testing — at continuously increasing workload
- Stress Testing — at or beyond the limits of the anticipated workload
- Endurance Testing — at continuous and significant workload
- Spike Testing — at suddenly and substantially increased workload

Use Case Testing

- The most widely used testing technique, followed by exploratory testing.
- Use case describes how a system will respond to a given scenario created by the user.
- It is user-oriented and focuses on the actions and the actor, not taking into account the system input and output.
- Developers write use cases and the behaviour of the system is tested accordingly.
- Testers, in their turn, use them to create test cases.
- Use case testing checks whether the path used by the user is working as intended and makes sure the tasks can be accomplished successfully.
- Applying use case testing, analysts can detect shortcomings and modify the system so that it attains efficiency and accuracy.

Exploratory Testing

- The exploratory testing technique was first described by Cem Kaner. a software engineering professor and consumer advocate, as

"***a style of software testing that emphasizes the personal freedom and responsibility of the individual tester to continually optimize the value of her work by treating test-related learning, test design, test execution, and test result interpretation as mutually supportive activities that run in parallel throughout the project*.**"

- Using the ad hoc method, exploratory testing does not rely on predefined and documented test cases and test steps as most testing types do.
- Instead, it is an interactive and free-form process, with the main focus on validating user experience, not code.
- It has much in common with the ad hoc or intuitive testing but is more systematic.
- Applying exploratory testing, skilled testers can provide valuable and auditable results.

Usability Testing

- usability testing is performed from the end user’s perspective to see if the system is easy to use.
- Unlike user acceptance testing this verifies that the final product meets the set requirements

Types of testing: manual and automation

Manual Testing:

- Manual testing is the process of testing the software manually to find the defects.
- Tester should have the perspective of end users and to ensure all the features are working as mentioned in the requirement document.
- Testers execute the test cases and generate the reports manually without using any automation tools.

Automation Testing:

- Speeds up and improve the quality of software testing
- In terms of continuous testing it eases the burden of managing all of the testing needs, allowing more time and effort to be spent on creating effective test cases.
- automated testing tools including Selenium, TestComplete, and Ranorex.

The process of test automation is typically conducted in several consecutive steps:

- Preliminary Project Analysis
- Framework Engineering
- Test Cases Development
- Test Cases Implementation
- Iterative Framework Support

Benefits of test automation:

- Automation can be applied to almost every testing type, at every level.
- Automation minimizes the human effort required to efficiently run tests.
- Reduces time-to-market and the cost of errors because the tests are performed up to 10 times faster.
- Is more efficient as the framework covers over 90 percent of the code, unveiling the issues that might not be visible in manual testing and can be scaled as the product grows.

Regression Testing

- Regression testing is the practice of verifying software behaviour after updates to ensure that the changes haven’t impacted existing system functions, stability, and overall integrity.
- Regression testing can be applied to all levels and with all types of testing procedures but the most common way is to run regression testing according to use cases.
- Regression quality assurance workflow can be automated to avoid repetitive manual tests after each update.
- There are multiple regression testing techniques:
    - Retesting all test cases
    - Selecting specific test cases
    - Prioritizing test cases to verify the most critical ones first and then test the rest
    - Hybrid techniques

Tools to Automate Manual Testing

- Selenium
- QTP
- Jmeter
- Loadrunner
- TestLink
- Quality Center(ALM)

New subjects expected to affect software testing in near future are,

Security

- Network security
- System software security
- Client-side application security
- Server-side application security

Artificial Intelligence

- Testing solutions have unlimited number of situations requiring artificial intelligence to test them thoroughly.

Big Data

- Two major concerns regarding test data management are data compliance and big data.
- Big data testing is aimed at checking the quality of data and verifying data processing.
- Data quality check involves various characteristics like conformity, accuracy, duplication, consistency, validity, data completeness, etc.