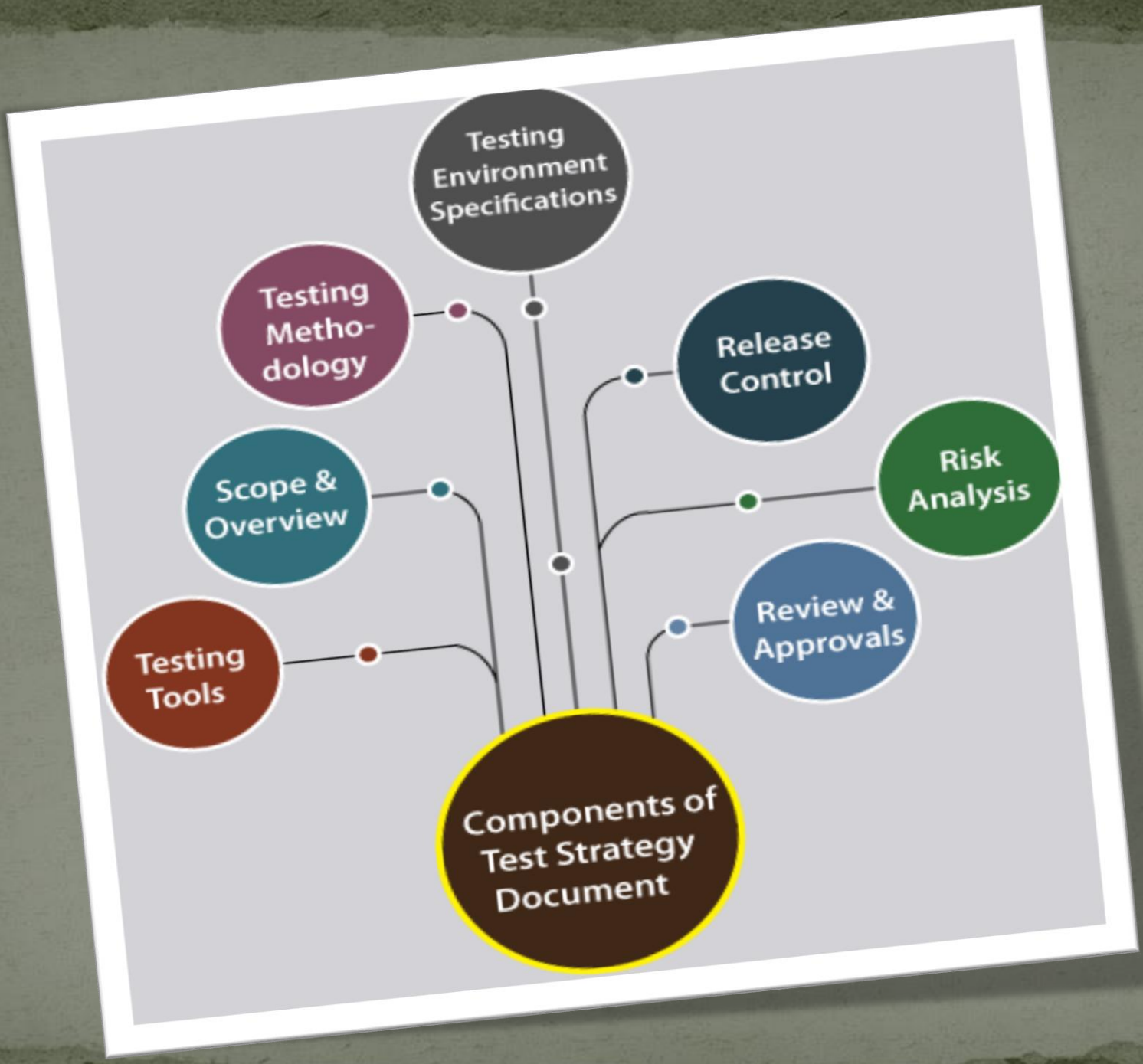
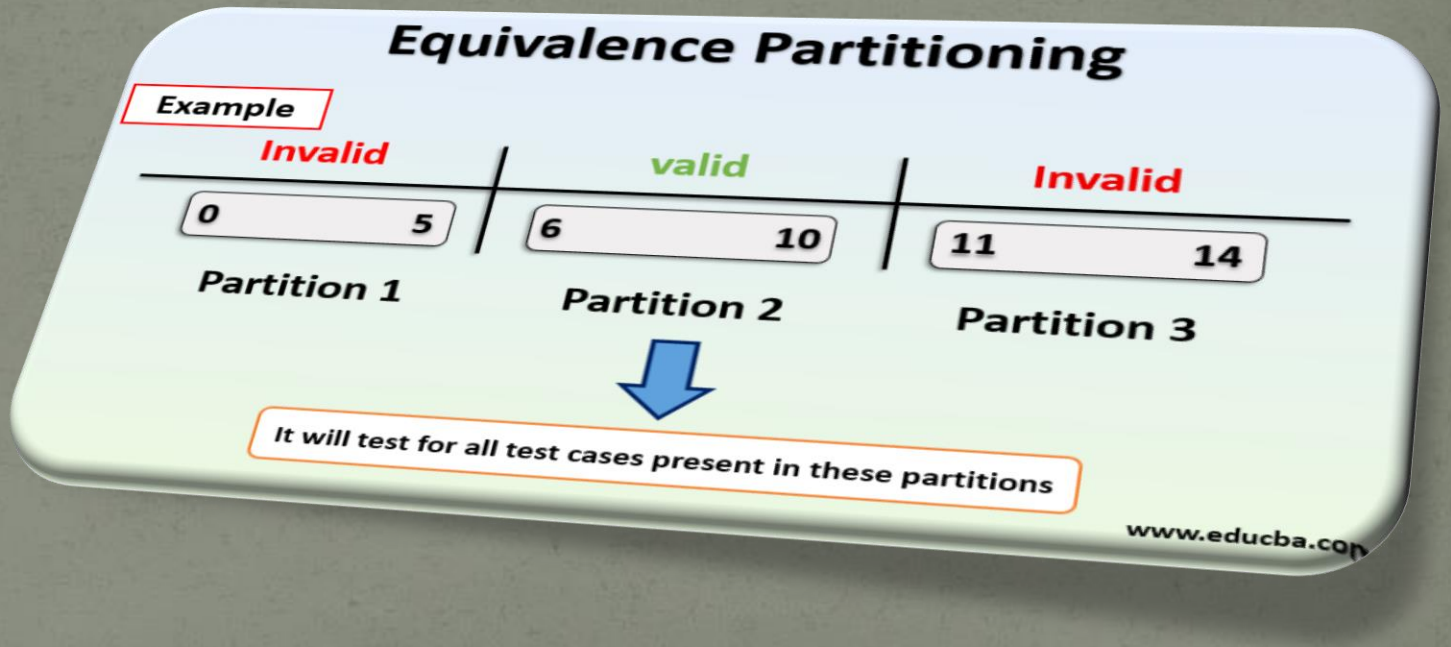
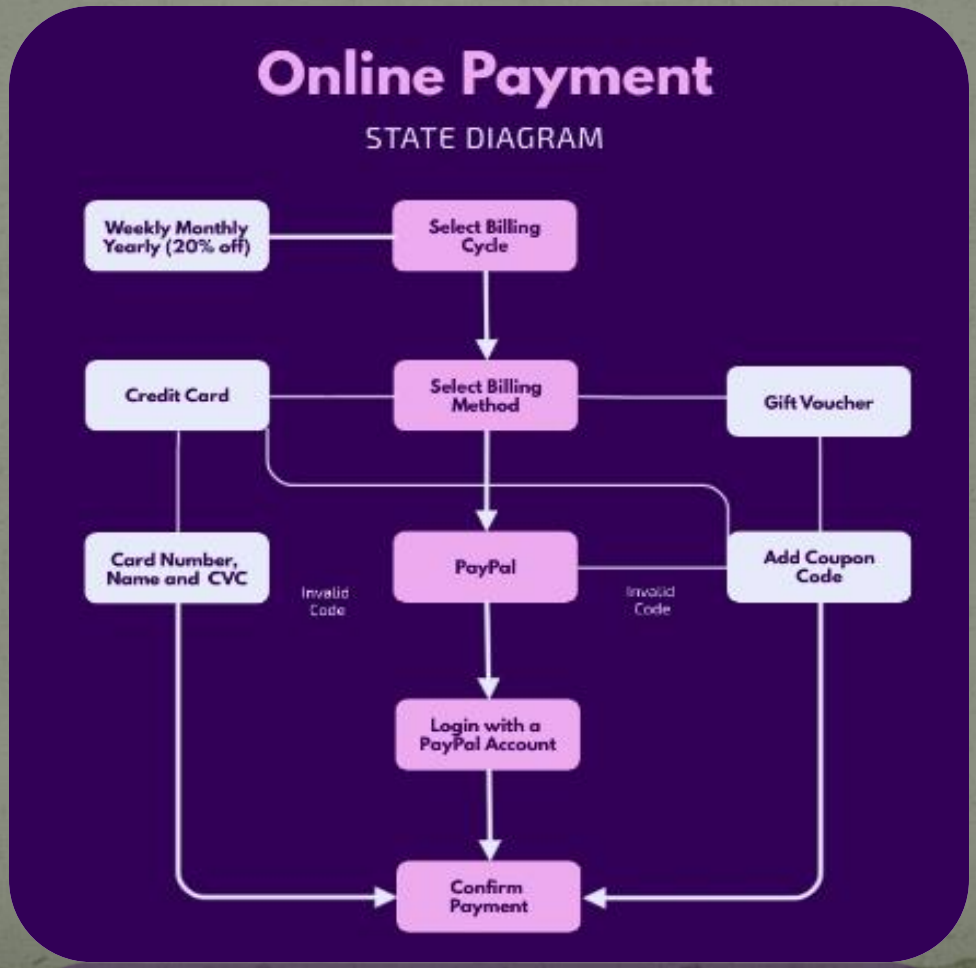
- Generating Test Architecture
- Test architecture refers to the design and structure of the testing framework and infrastructure in software testing. It encompasses the overall approach, components, and organization of the testing process.
- Some Considerations for Generating a Test Architecture
- **Test Strategy:** Define the overall testing approach, goals, and objectives. Determine the scope of testing, types of testing to be performed (such as functional, performance, security, etc.), and the target platforms or environments.


Testing Environment Specifications
Testing Metho-dology
Release Control
Scope & Overview
Risk Analysis
Testing Tools
Review & Approvals
Components of Test Strategy Document

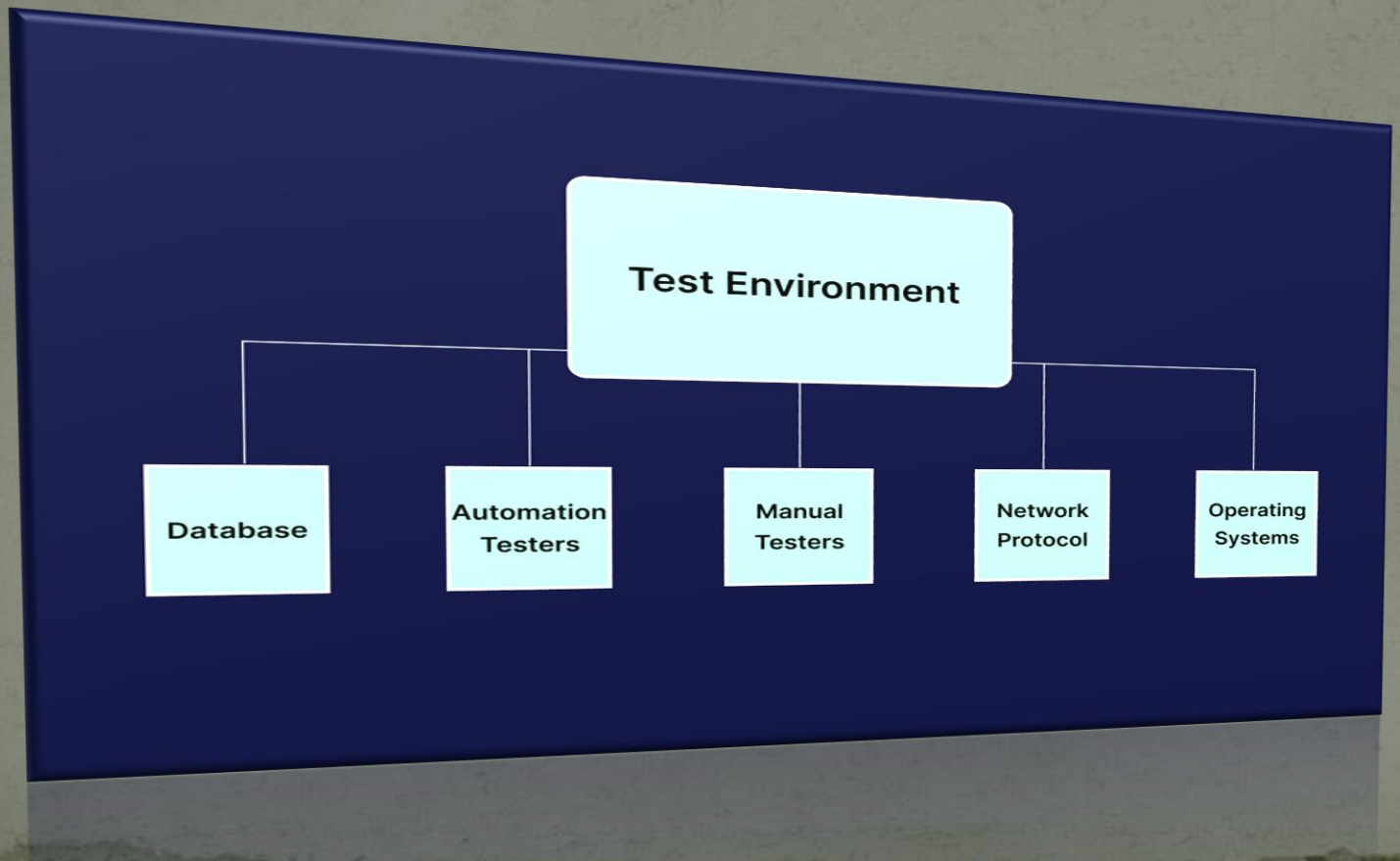
- Test Design Techniques: Select appropriate test design techniques, such as equivalence partitioning, boundary value analysis, decision tables, or state transition diagrams. These techniques help in identifying test cases and test scenarios to achieve sufficient test coverage.



- A state transition diagram is used to represent a finite state machine. These are used to model objects which have a finite number of possible states and whose interaction with the outside world can be described by its state changes in response to a finite number of events.

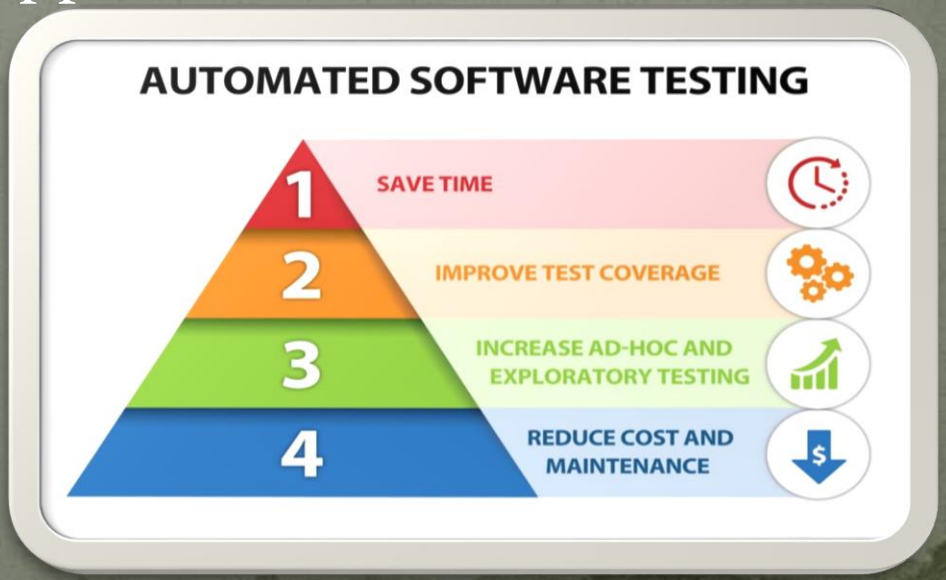


- Test Environment: Set up the necessary test environment, including hardware, software, and network configurations, to replicate (نقل)the production environment as closely as possible. Consider the availability of test servers, databases, and test data(https://www.kaggle.com/).



- Test Data Management: Plan how test data will be created, sourced, and managed. Define strategies for generating realistic (حقیقت پسندانہ)and representative test data that covers various scenarios (منظر نامے) and edge (کنارے) cases. Consider data privacy and security requirements.
- Data privacy is about proper usage, collection, retention, deletion, and storage of data. Data security is policies, methods, and means to secure personal data.

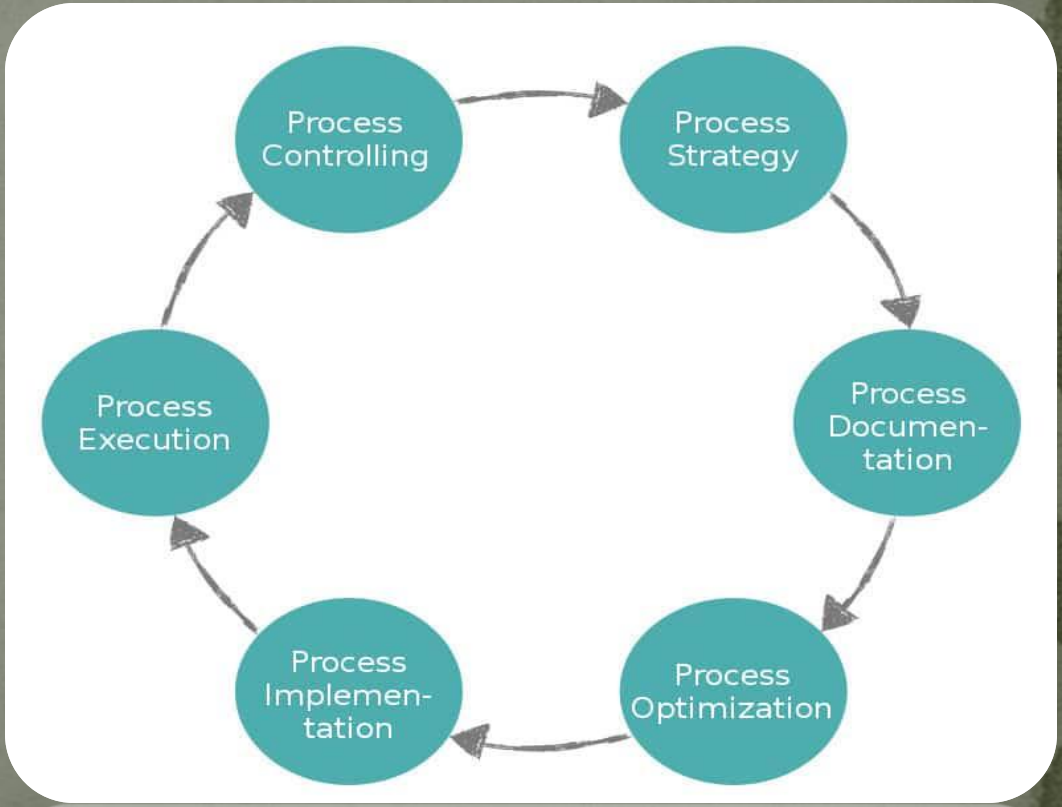

- Test Automation: Decide on the level of test automation needed and select appropriate automation tools and frameworks. Determine which tests can be automated, such as regression tests or repetitive tasks, to improve efficiency and coverage.
  - **Selenium** (Web Application Testing)
  - **Appium** (Mobile Application Testing)
  - **PyTest** (Python-based Testing Framework )
  - **Junit** (Framework for Java Applications)



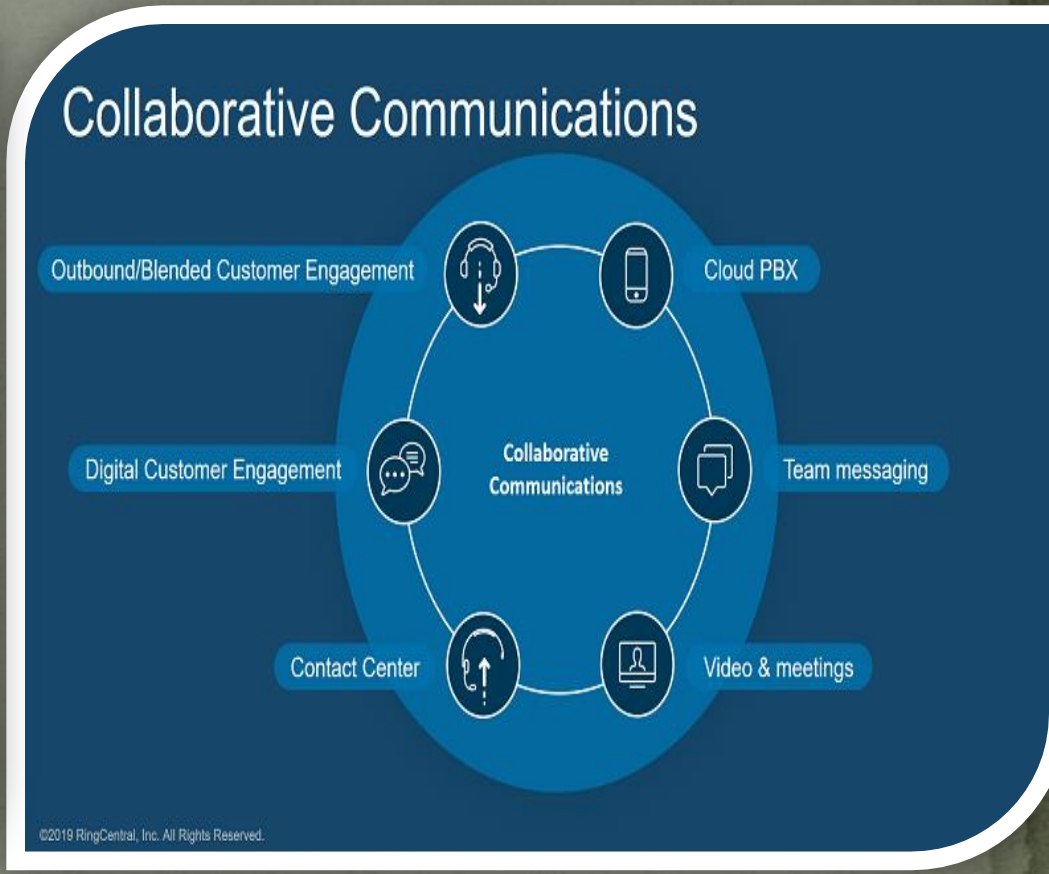
- Test Case Management: Establish a systematic approach for managing test cases. Define how test cases will be created, organized, and maintained. Utilize test case management tools to track the execution, results, and coverage of test cases.


- Test Execution and Reporting: Determine how tests will be executed and monitored. Establish test execution schedules, test environments, and test data dependencies. Define metrics and reporting mechanisms to track test progress and provide visibility to stakeholders متعلقين. Visit on internet sites

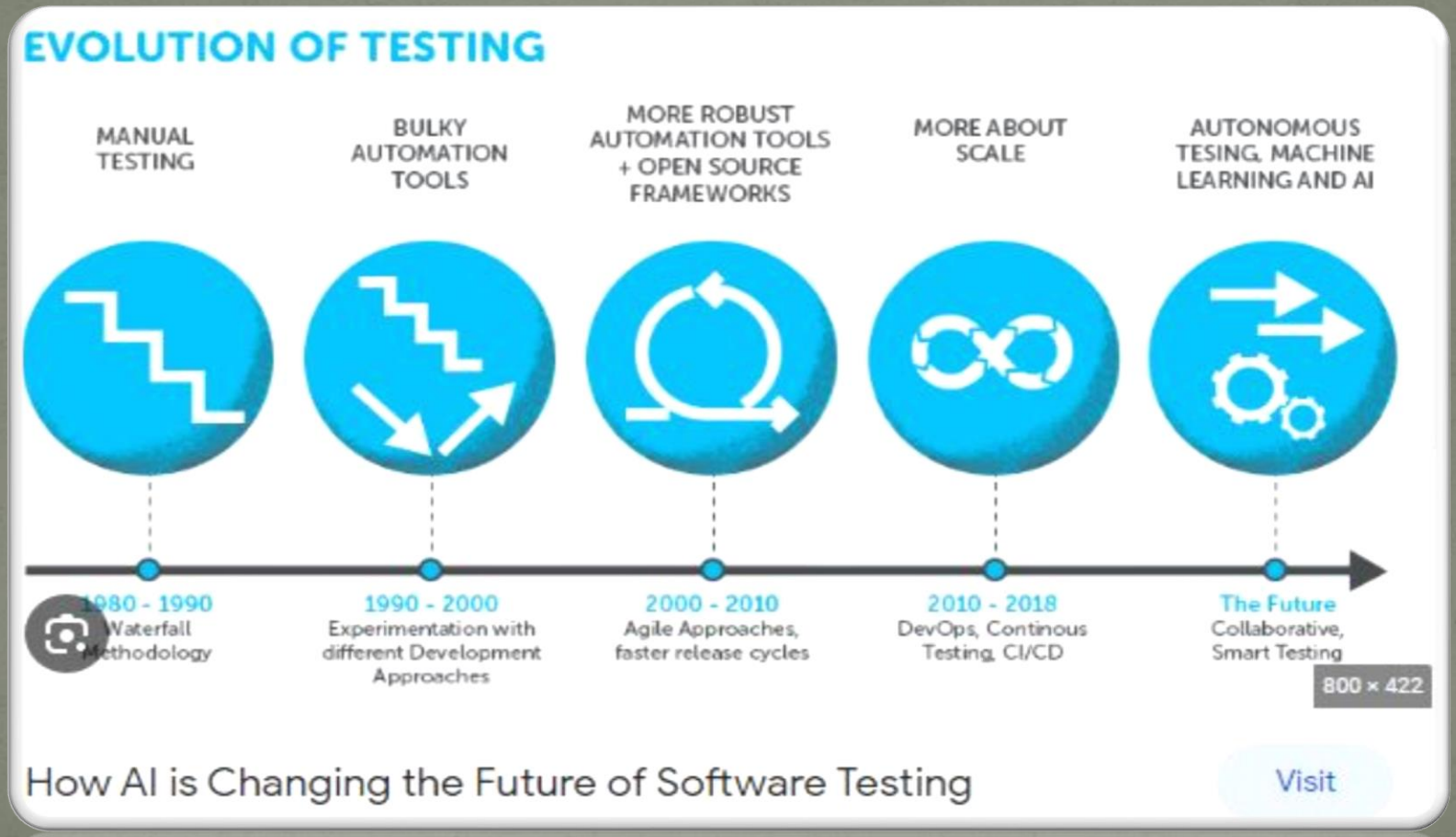
- Defect Management: Define the process for capturing, tracking, and resolving defects. Establish guidelines for defect logging, prioritization ترجيح, and resolution. Utilize defect tracking tools to effectively manage the defect lifecycle.



- Collaboration and Communication: Plan for effective collaboration and communication among the testing team and other stakeholders. Define channels and tools for sharing test artifacts نمونے, progress updates, and issue resolution.



- Continuous Improvement: Establish mechanisms to collect feedback and continuously improve the test architecture. Analyze test results, identify areas of improvement, and incorporate lessons learned into future testing cycles.



- Pre/Post Processing: Pre-processing and post-processing are important steps in software testing that help ensure the accuracy and efficiency of the testing process.
- Pre-processing: refers to the activities and steps performed before executing the actual tests. These activities are crucial for preparing the testing environment, test data, and test cases.
- Some key Pre-Processing Activities

1. Test Planning
2. Requirement Analysis
3. Test Environment Setup
4. Test Data Preparation
5. Test Case Development
6. Test Scripting/Automation
7. Test Setup and Configuration
8. Test Data and Environment Sanity Checks

- Test Planning: This involves defining the testing objectives, scope, and strategy. It includes identifying the test levels, test types, and prioritizing the features or functionalities to be tested. Test planning also involves estimating the resources, time, and effort required for testing.
- Requirement Analysis: Understanding the software requirements and specifications to derive testable scenarios and identify potential risks. This includes reviewing the requirements, discussing with stakeholders, and clarifying any ambiguities or inconsistencies.
- Test Environment Setup: Creating a suitable test environment that mirrors the production environment as closely as possible. This may involve setting up hardware, software, network configurations, databases, and other dependencies necessary for the software to run.

- Test Data Preparation: Identifying or creating the necessary test data to cover different scenarios and conditions. This includes creating valid and invalid data, boundary values, and edge cases. Test data may be generated manually or using automated tools.
- Test Case Development: Writing test cases based on the requirements and test scenarios identified during the requirement analysis phase. Test cases should be comprehensive, covering all relevant functionalities, inputs, and expected outputs. Each test case should have clear steps, expected results, and any preconditions or prerequisites.
- Test Scripting/Automation: If test automation is being used, this involves developing test scripts or test automation code using appropriate automation tools and frameworks. Test scripts automate the execution of test cases, reducing manual effort and increasing efficiency.

- Test Setup and Configuration: Configuring the testing tools and frameworks, including setting up test environments, databases, mock services, and any required integrations or dependencies.
- Test Data and Environment Sanity Checks: Verifying the correctness and integrity of the test data and environment. This involves ensuring that the test data is correctly mapped to the test cases and that the test environment is properly set up and ready for testing.

- Post-processing: Post-processing refers to the activities and steps performed after executing the tests. These activities involve analyzing the test results, reporting and tracking defects, and summarizing the testing effort.
- some key post-processing activities
- Test Result Analysis
- Defect Reporting
- Defect Tracking
- Defect Retesting
- Test Closure
- Test Metrics and Analysis
- Test Documentation

- Test Result Analysis: Analyzing the test results to determine if the software meets the expected criteria. This involves comparing the actual results with the expected results and identifying any discrepancies or deviations. Test result analysis helps in identifying defects or issues that need to be addressed.
- Defect Reporting: Documenting and reporting any issues or defects found during the testing process. This includes capturing detailed information about each defect, such as its description, steps to reproduce, severity, priority, and any supporting attachments or screenshots. Effective defect reporting ensures that developers can understand and reproduce the issues accurately.

- Defect Tracking: Tracking the reported defects to ensure they are addressed by the development team. This involves assigning the defects to the appropriate stakeholders, monitoring their progress, and updating the status as the defects are fixed and verified. Defect tracking helps in maintaining transparency and accountability throughout the defect resolution process.
- Defect Retesting: After a defect is fixed, retesting is performed to verify that the fix has resolved the issue and has not introduced any new defects. This involves executing the specific test cases related to the fixed defect and validating the expected behavior.
- Test Closure: Summarizing the testing effort and providing closure to the testing phase. This includes preparing test closure reports that capture essential information about the testing activities, such as the test coverage, test execution details, defect statistics, and any unresolved issues. Test closure reports also provide recommendations for future testing

- Test Metrics and Analysis: Collecting and analyzing various metrics related to the testing effort. This may include metrics such as test coverage, defect density, test case execution status, and test efficiency. Test metrics help in evaluating the effectiveness and efficiency of the testing process and can provide insights for process improvement.
- Test Documentation: Updating the test documentation, including test cases, test data, and any test-related artifacts. This ensures that the documentation reflects the latest changes made during the testing process and serves as a reference for future testing efforts.


QA process stages
Requirements analysis
Test planning
Test design
Test execution and defect reporting
Retesting and testing
Release testing

- **Test Maintenance and Job Specific Metrics**

Test maintenance and job-specific metrics are important aspects of software testing that help measure the effectiveness, efficiency, and quality of testing efforts. These metrics provide insights into the progress, performance, and areas of improvement for the testing process and the testing team.

Test Case Effectiveness

Test Case Execution Productivity

Test Case Failure Rate

Test Execution Coverage

Defect Detection Rate

Defect Leakage

Test Maintenance Effort

Test Automation Coverage

Test Execution Time

Test Case Reusability

**Test Case Effectiveness:** This metric measures the percentage of test cases that find defects. It helps evaluate the quality and relevance of test cases. A high test case effectiveness indicates that a significant number of defects are being found, while a low effectiveness may indicate the need for test case refinement.

**Test Case Execution Productivity:** It measures the efficiency of executing test cases. This metric considers the number of test cases executed within a specific timeframe or against a specific target, such as test cases executed per hour or per day. It helps assess the productivity of the testing team.

**Test Case Failure Rate:** This metric indicates the percentage of test cases that fail during execution. It helps identify the stability and quality of the software under test. A high failure rate may suggest underlying issues in the application, while a low failure rate indicates a more stable

Test Execution Coverage: It measures the extent to which the software has been tested. This metric considers the percentage of requirements, functionalities, or code covered by executed test cases. It helps assess the completeness of the testing effort and identifies any gaps in test coverage.

Defect Detection Rate: This metric measures the number of defects identified during testing, typically per unit of time. It helps gauge the effectiveness of the testing process in finding and reporting defects. A higher defect detection rate suggests better defect identification and reporting.

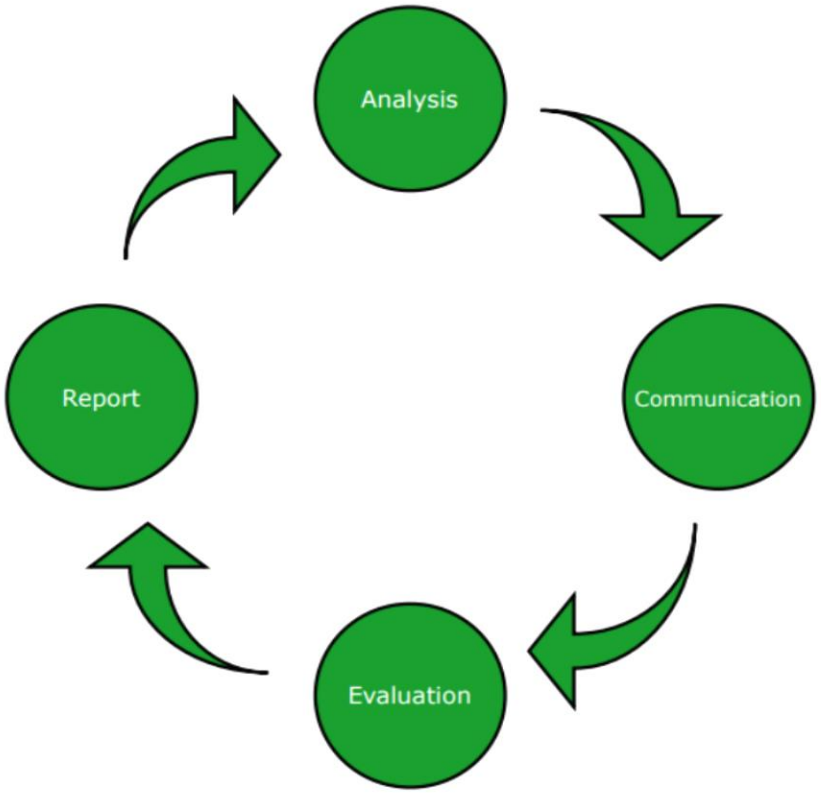
**Defect Leakage:** It represents the percentage of defects that escape the testing phase and are found by end-users or customers. This metric helps assess the effectiveness of the testing process in preventing defects from reaching production. A lower defect leakage rate indicates a more robust testing process.

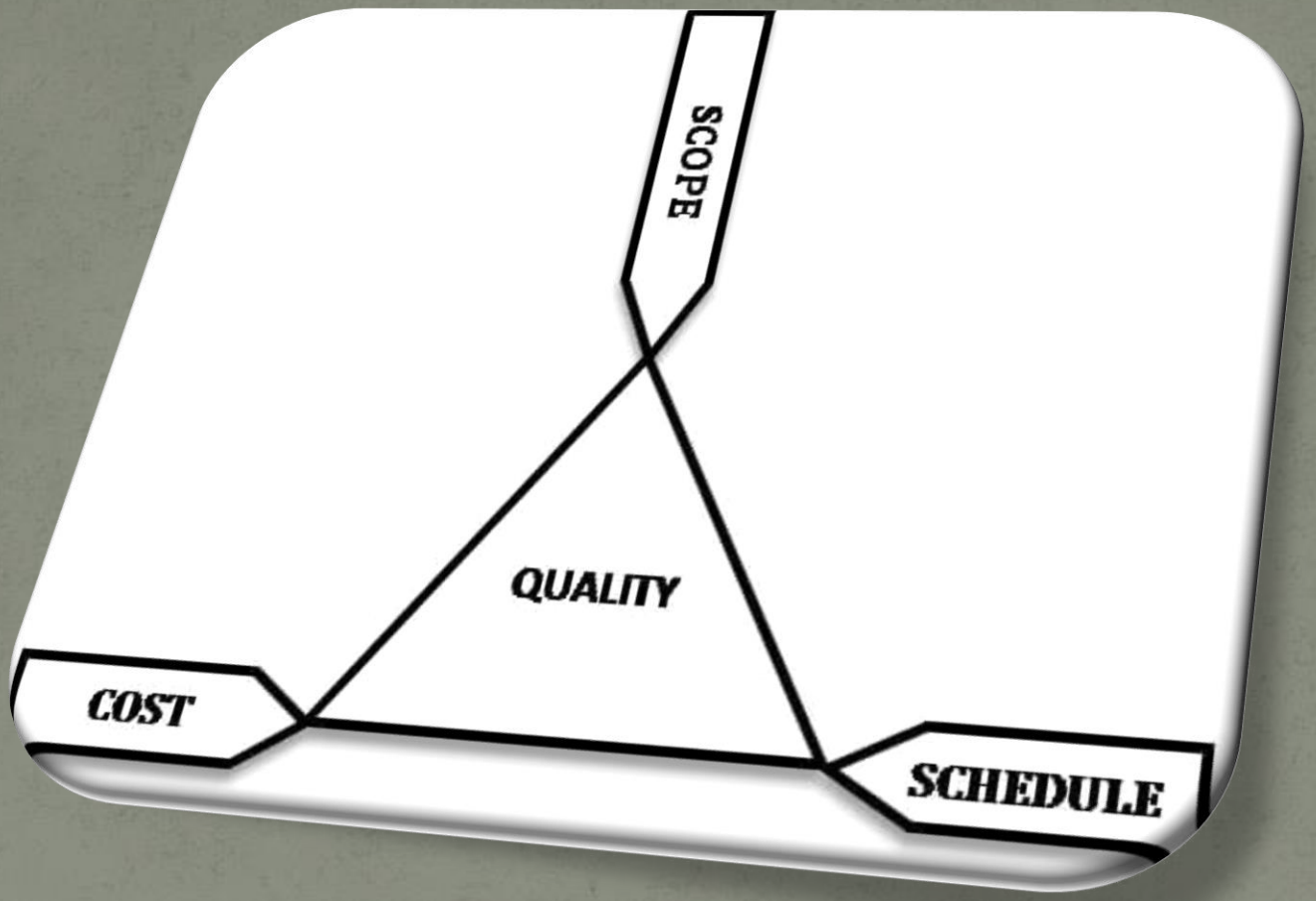
**Test Maintenance Effort:** This metric measures the effort required to maintain and update test assets, such as test cases, test data, and test environments. It includes activities like test case maintenance, test environment setup, and test data management. Tracking this metric helps identify resource allocation and optimization opportunities.

**Test Automation Coverage:** It measures the percentage of test cases automated compared to the total number of test cases. This metric helps assess the level of test automation in the testing process. Higher automation coverage reduces manual effort, increases efficiency, and improves test execution speed.

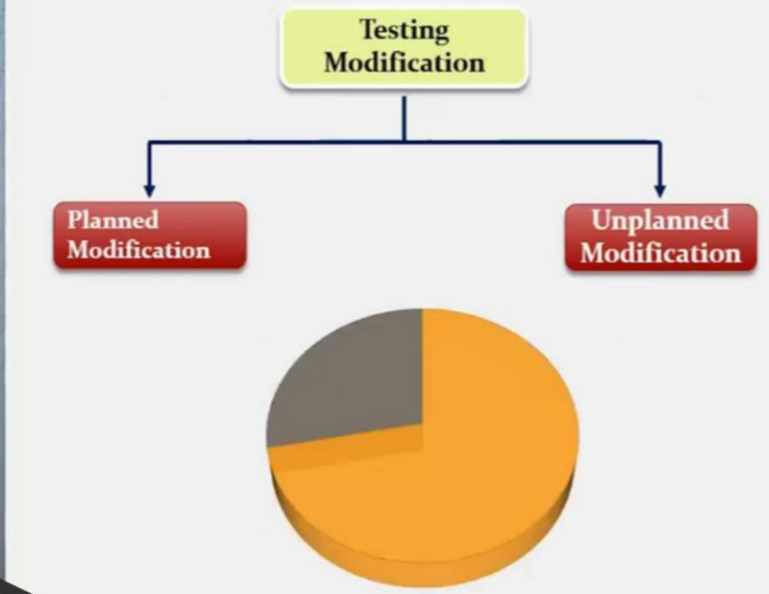
**Test Execution Time:** This metric measures the time taken to execute a set of test cases. It helps identify any performance bottlenecks in the testing process and provides insights into test execution efficiency.

**Test Case Reusability:** It measures the percentage of test cases that can be reused across different test cycles, releases, or projects. This metric highlights the effectiveness of test case design and the potential for reducing redundancy and improving efficiency.


Analysis
Communication
Evaluation
Report


SCOPE
QUALITY
COST
SCHEDULE

# Maintenance Testing cont.



# Any Question.....?