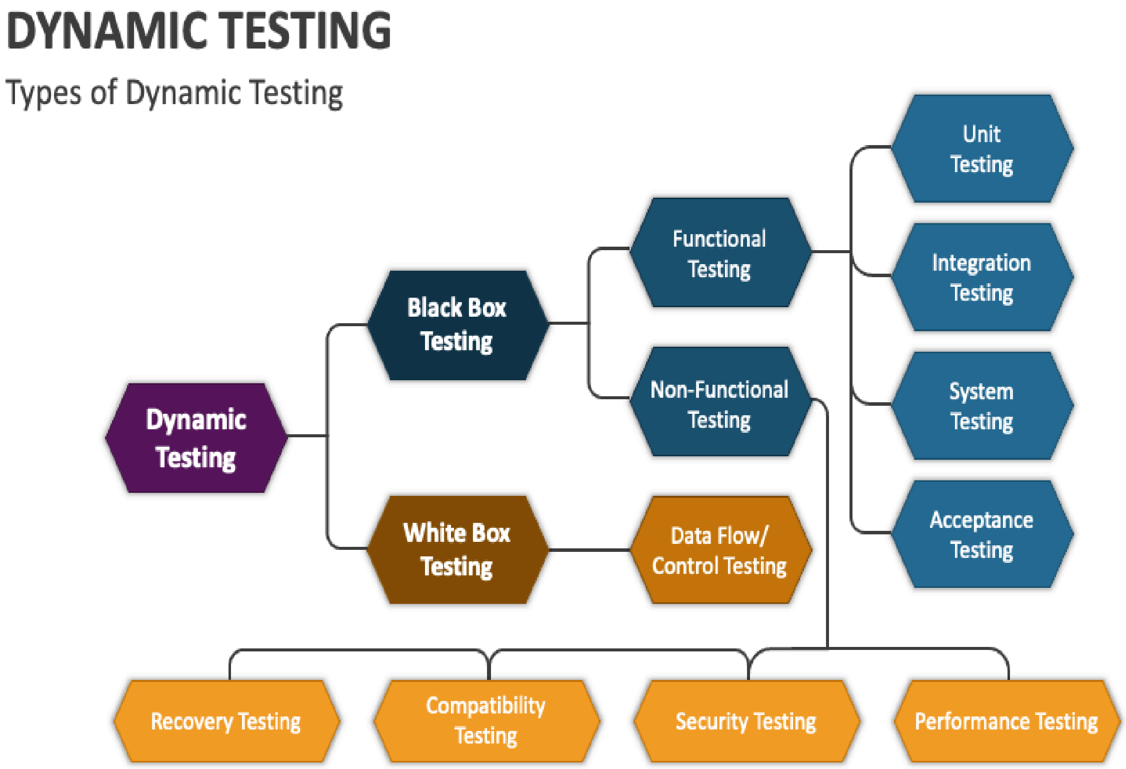
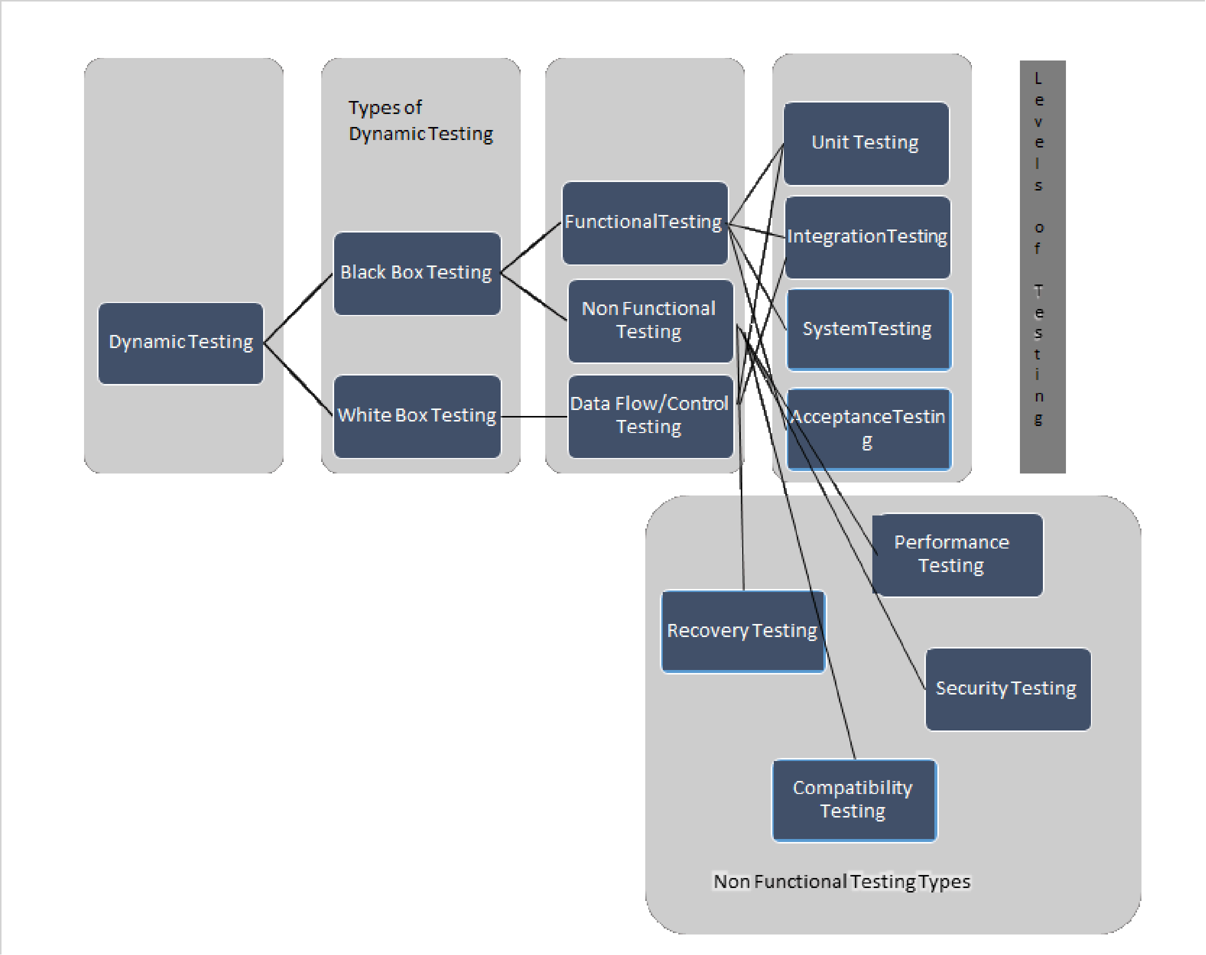
Types of Dynamic Testing

Dynamic Testing is classified into two categories

- White Box Testing
- Black Box Testing





White-box testing and black-box testing are two common approaches used in software testing, each with its own methodologies and objectives.

White-box Testing:

- White-box testing, also known as clear-box testing, glass-box testing, or structural testing, involves testing the internal structure of the software code.
- Testers have access to the internal workings of the software, including the source code, algorithms, and architecture.
- The primary goal is to validate the correctness of the code logic, ensure that all code paths are tested, and verify that the code adheres to design specifications.
- Techniques used in white-box testing include statement coverage, branch coverage, path coverage, and condition coverage.
- White-box testing is typically performed by developers or specialised testers who have knowledge of the software's internal implementation.

Black-box Testing:

- Black-box testing, also known as functional testing, opaque-box testing, or behavioural testing, focuses on testing the software's functionality without knowing its internal structure.
- Testers do not have access to the source code or the implementation details; they interact with the software as an external user would.
- The primary goal is to verify that the software behaves as expected based on its specifications, requirements, and user inputs.
- Test cases are designed based on the software's functional requirements, use cases, and specifications.
- Techniques used in black-box testing include equivalence partitioning, boundary value analysis, decision table testing, and state transition testing.
- Black-box testing is often performed by independent testers who simulate various inputs and scenarios to evaluate the software's behaviour.

In summary, white-box testing focuses on the internal logic and structure of the software, while black-box testing focuses on the software's external behavior and functionality. Both approaches are essential for ensuring the quality and reliability of software systems.

Dynamic testing involves assessing software behaviour by executing it with specific inputs and observing its outputs. There are several types of dynamic testing techniques used to evaluate different aspects of software quality. Here are some common types of dynamic testing:

Functional Testing: Functional testing verifies that the software functions according to its specifications and requirements. It focuses on testing the functionality of the software, including user interactions, data processing, and business logic. Functional testing ensures that the software performs the intended tasks correctly.

Integration Testing: Integration testing verifies the interactions between different modules or components of the software. It tests the interfaces and interactions between integrated units to ensure they work together seamlessly. Integration testing can be performed at various levels, including module-to-module, subsystem-to-subsystem, and system-to-system integration.

System Testing: System testing evaluates the complete and integrated software system to ensure it meets its specified requirements. It involves testing the software as a whole, including its functionality, performance, reliability, and usability. System testing verifies that the software behaves as expected in different environments and scenarios.

Acceptance Testing: Acceptance testing validates whether the software meets the acceptance criteria and satisfies the stakeholders' requirements. It involves testing the software from the end-users' perspective to ensure it meets their needs and expectations. Acceptance testing can include alpha testing, beta testing, user acceptance testing (UAT), and other forms of customer validation.

Regression Testing: Regression testing verifies that recent code changes have not introduced new defects or unintended side effects into the software. It involves rerunning previously executed test cases to ensure that existing functionality remains intact after modifications or updates. Regression testing helps maintain software quality and stability throughout the development lifecycle.

Load Testing: Load testing evaluates the performance and scalability of the software under expected load conditions. It involves subjecting the software to simulated user loads and assessing its response time, throughput, and resource utilisation. Load testing helps identify performance bottlenecks and ensure the software can handle expected levels of user traffic.

Stress Testing: Stress testing assesses the software's robustness and reliability by subjecting it to extreme conditions beyond its normal operating limits. It involves applying high loads, excessive inputs, or adverse environmental conditions to test the software's resilience and ability to recover from failures.

Security Testing: Security testing assesses the software's ability to protect data, prevent unauthorised access, and resist attacks. It involves identifying and addressing security vulnerabilities such as authentication flaws, authorization issues, encryption weaknesses, and injection attacks.

These dynamic testing types are essential for evaluating different aspects of software quality, performance, and reliability, helping ensure the delivery of a high-quality product to end-users.