Introduction and Review of Software Testing

Course Code: CIT-5202

Title: Software Testing & Quality Assurance

Course Teacher:

Md. Atikqur Rahaman

Dept. of CSIT, Faculty of CSE,

PSTU, Bangladesh.

Objective

The objective of this presentation is to show the

- How to define Software Testing Principles
- What are the types of Software Tests
- What is Test Planning
- Test Execution and Reporting
- Real-Time Testing

How to define Software Testing Principles

- Testing

  The execution of a program to find its faults
- Verification

  The process of proving the programs correctness.
- Validation

  The process of finding errors by executing the program in a real environment
- Debugging

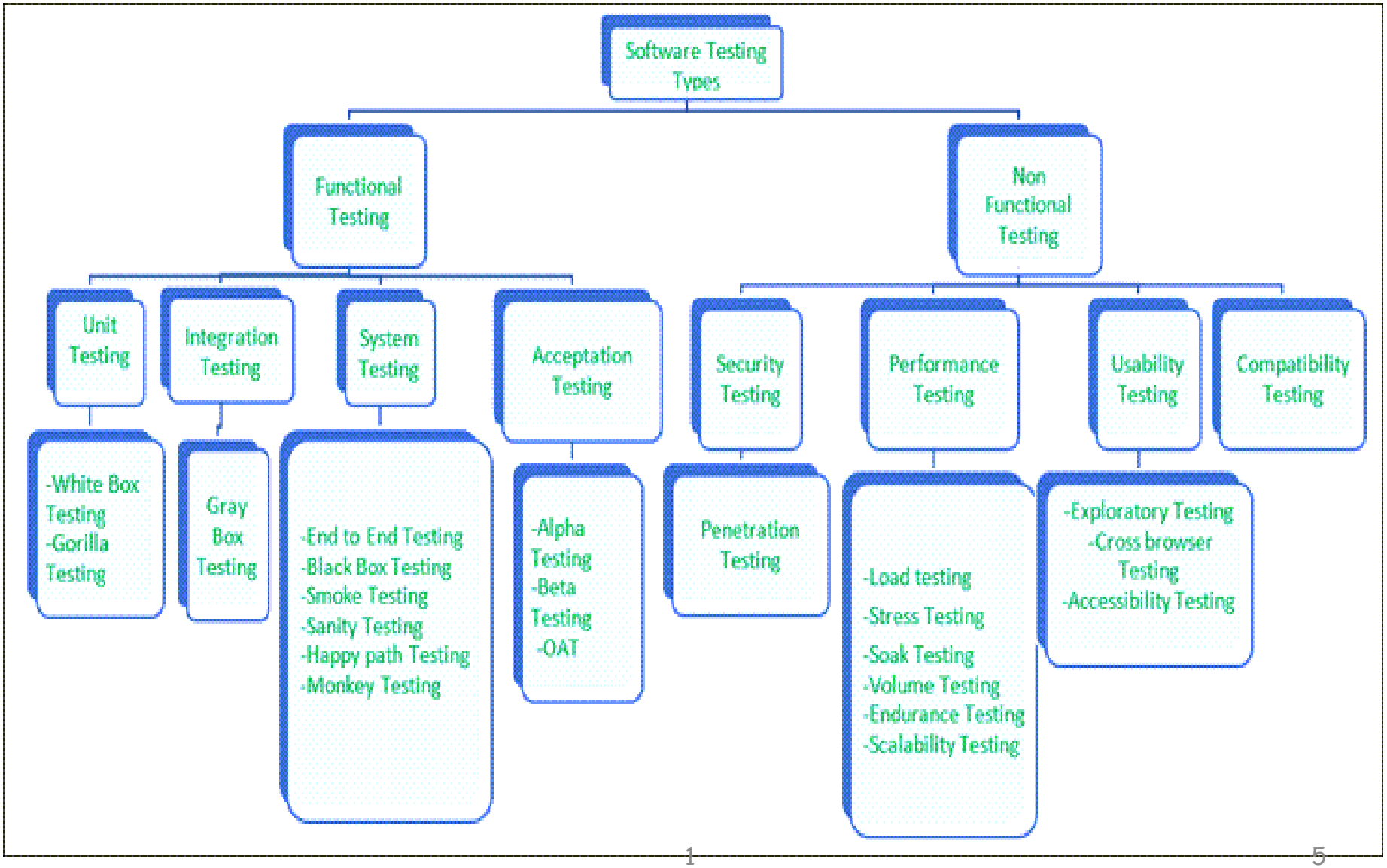
  Diagnosing the error and correct it

Software Testing Principles

The (07) seven principles of testing

1. Testing shows the presence of defects, not their absence
2. Exhaustive testing is impossible
3. Early testing saves time and money.
4. Defects cluster together.
5. Beware of the pesticide paradox
6. Testing is context dependent.
7. Absence-of-errors is a fallacy.

Types of Software Tests



List of Software Tests

Functional testing

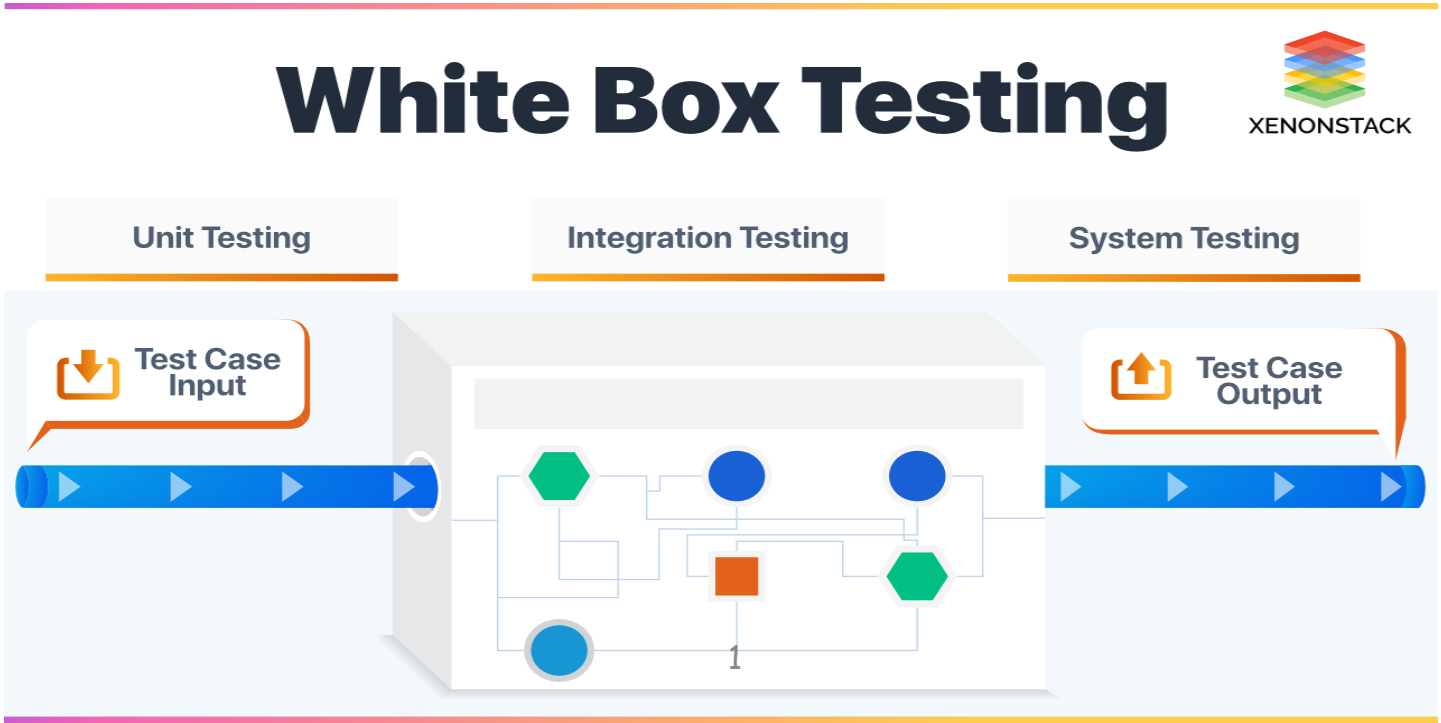
- Unit Testing (White Box)
- Integration Testing
- Function Testing (Black Box)
- Regression Testing
- System Test
- Acceptance and Installation Tests

Non-Functional testing:

- Security testing
- Performance testing
- Usability testing
- Compatibility testing

Unit Testing (White Box)

- White box testing is an approach that allows testers to inspect and verify the inner workings of a software system—its code, infrastructure, and integrations with external systems.
- Individual components are tested.
- It is a path test.
- To focus on a relatively small segment of code and aim to exercise a high percentage of the internal path



Unit Testing (White Box)

Benefits of Unit Testing

- Improve Quality and Performance. Unit testing can improve the quality of your codebase, making it more maintainable and less error-prone.
- Allows for Documentation.
- Find Bugs.
- Makes Debugging Easier.
- Reduces Software Complexity.

Unit Testing (White Box)

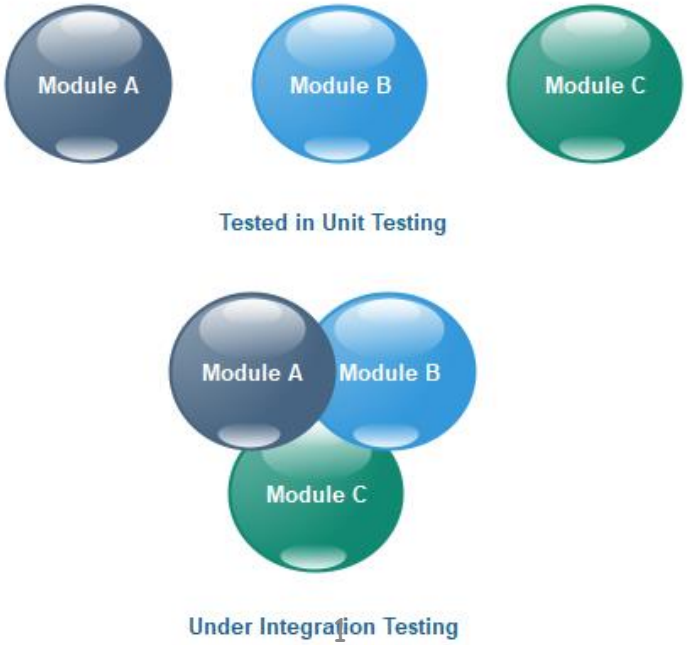
Disadvantages of Unit Testing

- With UT, you have to increase the amount of code that needs to be written. You usually have to write one or more unit tests depending on how complex things are.
- Unit tests are problematic when testing your user interface (UI). They are good for when you need to test business logic implementation but not great for UI.
- In comparison to those who say UT improves code, others say it makes it worse and ends up adding indirection that is pointless. Changing code and adding new code can mean navigational issues and more time spent before integration testing is even started.
- UT cannot and will not catch all errors in a program. There is no way it can test every execution path or find integration errors and full system issues.
- Unit tests have to be realistic. You want the unit you're testing to act as it will as part of the full system. If this doesn't happen, the test value and accuracy are compromised.

Integration Testing

- Integration testing -- also known as integration and testing (I&T) -- is **a type of software testing in which the different units, modules or components of a software application are tested as a combined entity**.

For example the fuel system may be tested in collaboration with an exhaust system, and later, these two module's working is tested in collaboration with the working of an engine. Now, this is integration testing.



Integration Testing

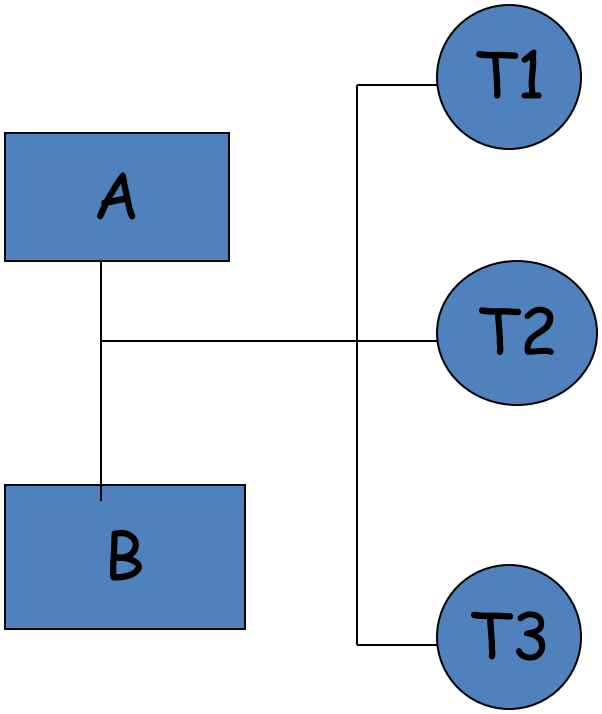
- Top-down Integration Test
- Bottom-up Integration Test
- Mixed/sandwich integration testing.
- Big-bang integration testing.

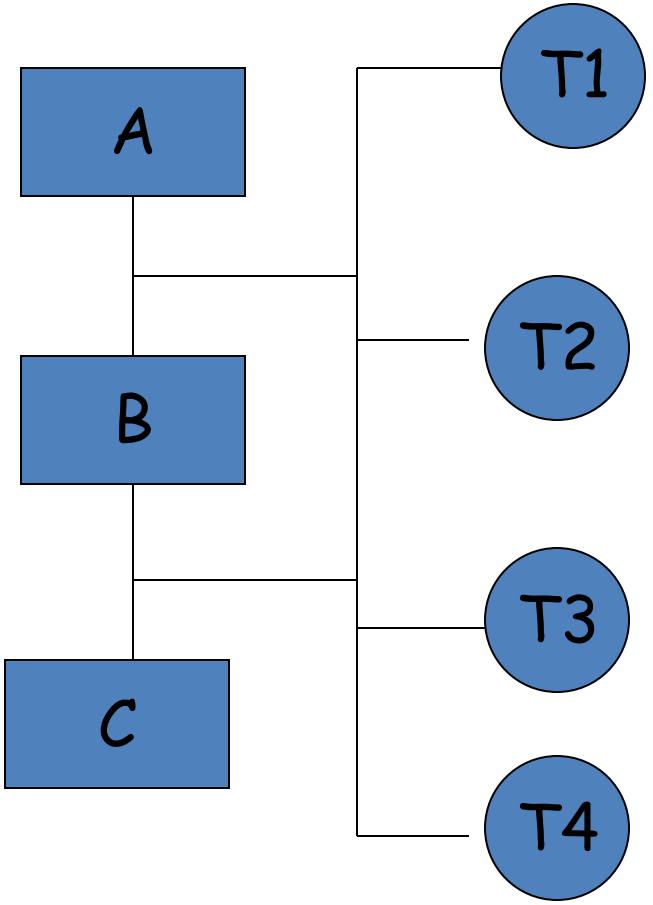
Integration Testing

Top-down Integration Test

- The control program is tested first. Modules are integrated one at a time. Emphasize on interface testing
- Advantages: No test drivers needed
- Interface errors are discovered early
- Modular features aid debugging
- Disadvantages: Test stubs are needed
- Errors in critical modules at low levels are found late.

Top-down Testing



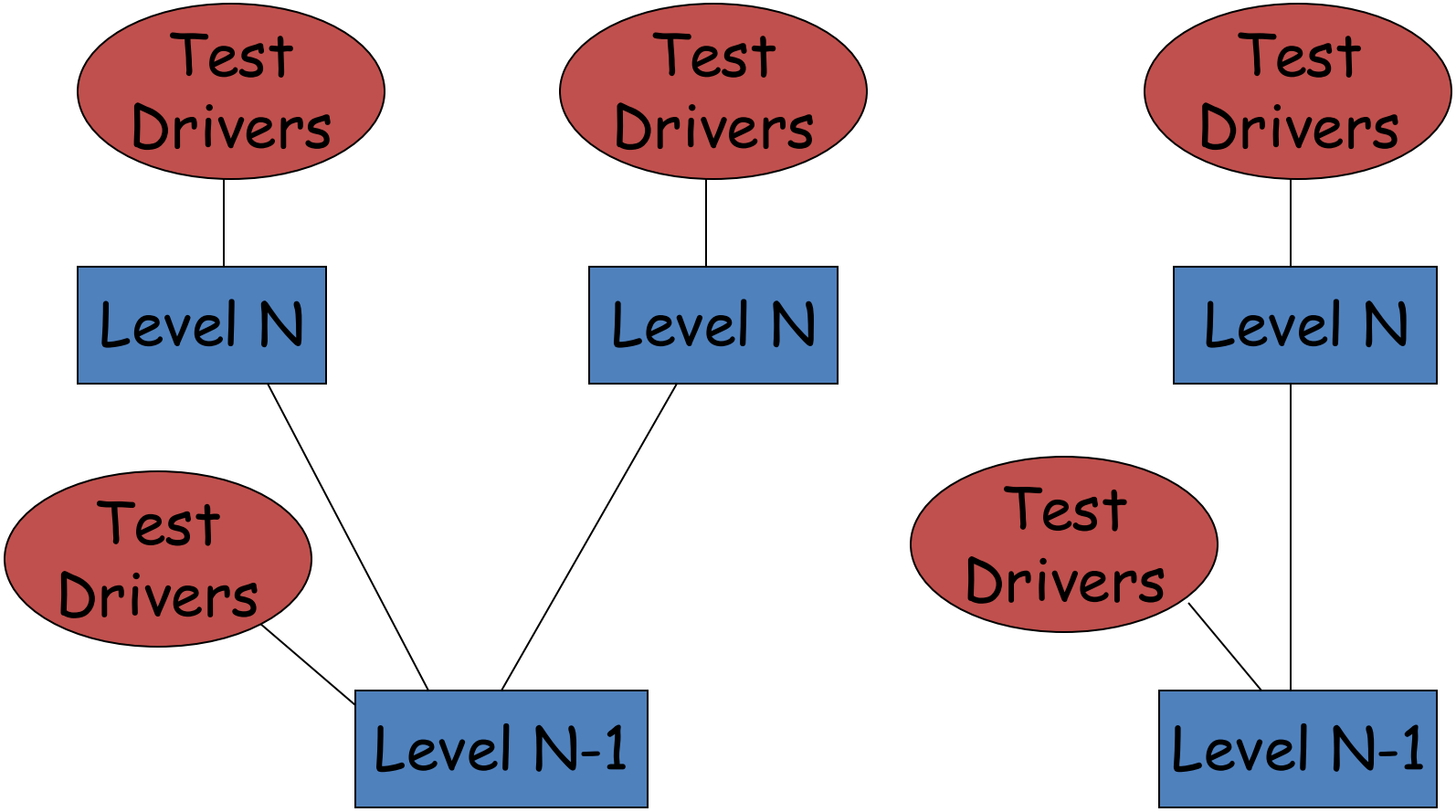


Integration Testing

Bottom-up Integration Test

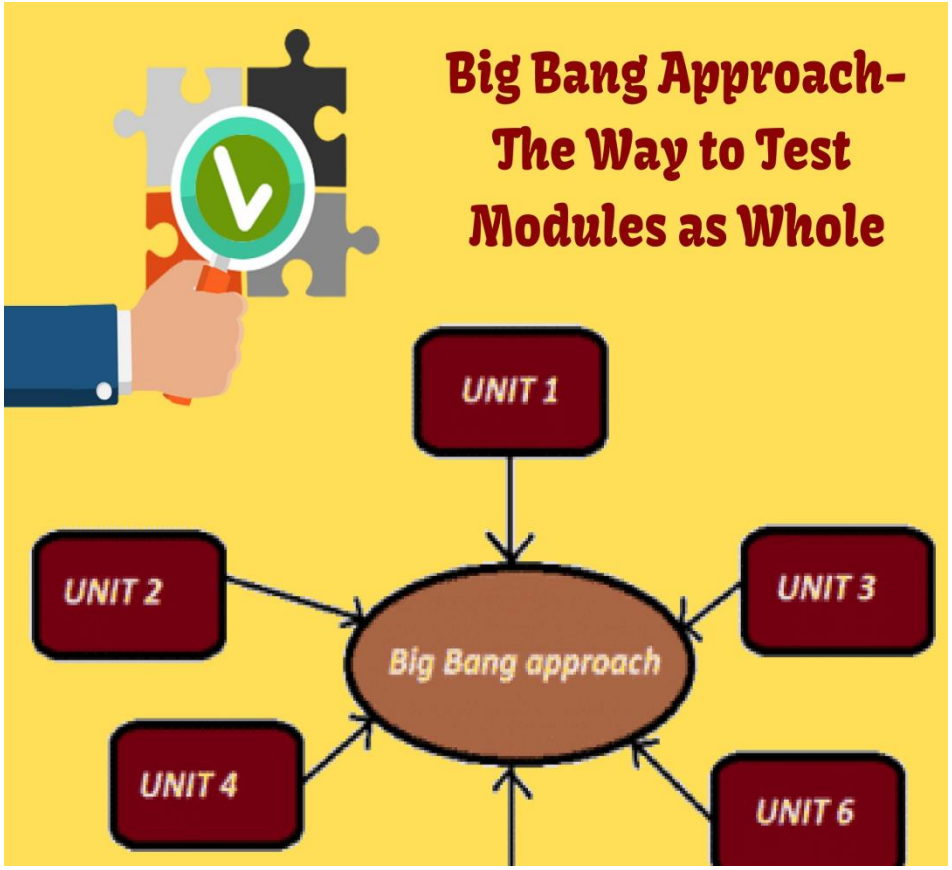
- Allow early testing aimed at proving feasibility
  Emphasize on module functionality and performance
- Advantages: No test stubs are needed
  Errors in critical modules are found early
- Disadvantages: Test drivers are needed
  Interface errors are discovered late

Bottom-up testing



Big-bang integration testing

Big bang integration testing is **a testing approach where all components or modules are integrated and tested as a single unit**. This is done after all modules have been completed and before any system-level testing is performed.



Big-bang integration testing

Advantage of Big Bang Integration:

- Big bang integration testing allows for testing of complex interactions between components. This is beneficial as it allows for the identification of errors that may not be detected by other testing methods

Disadvantages of Big Bang Integration:

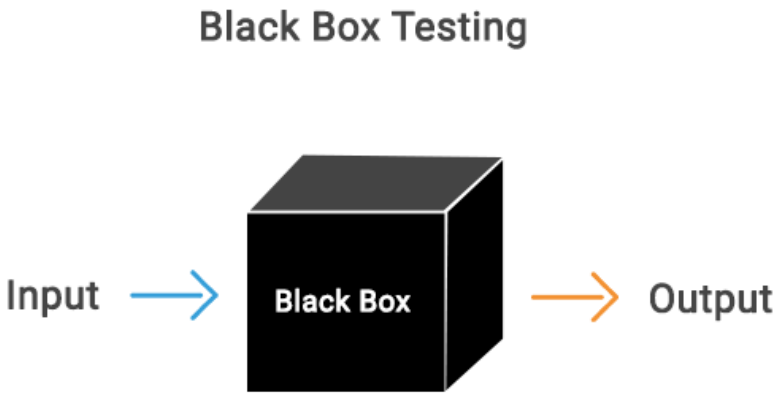
- In general **it is very time consuming**. It can be difficult to identify and fix errors that are discovered late in the testing process.

System/Function Testing (Black Box)

- **Black box testing** is a software testing methodology in which the tester analyzes the functionality of an application without a thorough knowledge of its internal design.

Example:

- we all have tried Black Box testing in our lives. For example, **while pressing the start button of a bike, we expect it to start without getting into its inner working mechanism**. In other words, it focuses on the functionality of the software without any need for coding knowledge



System/Function Testing (Black Box)

Advantage:

- Well suited and efficient for large code segments.
- Code access is not required.
- Clearly separates user's perspective from the developer's perspective through visibly defined roles
- Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language, or operating systems.

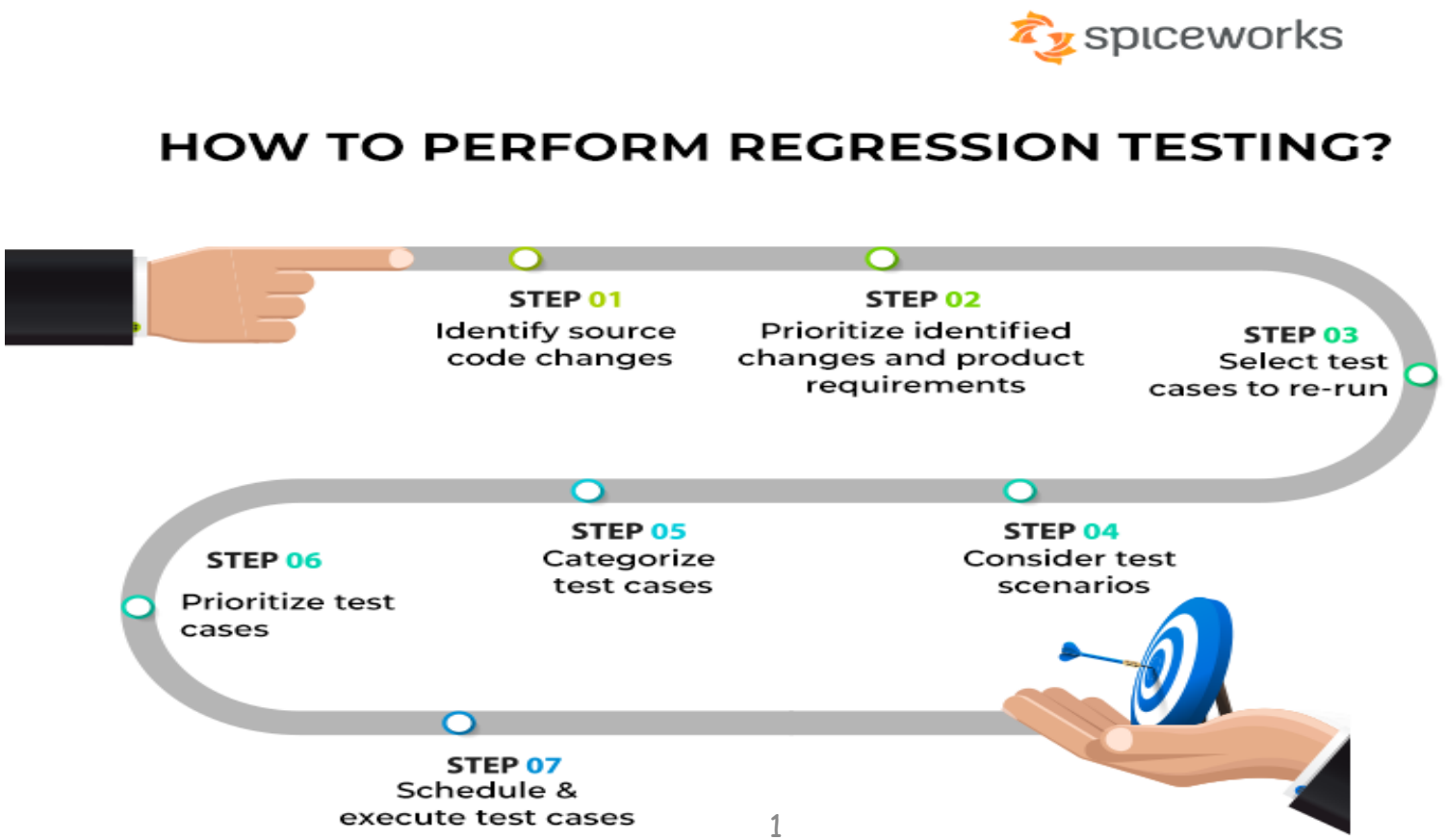
Disadvantage:

- Limited coverage, since only a selected number of test scenarios is actually performed.
- Inefficient testing, due to the fact that the tester only has limited knowledge about an application.
- Blind coverage
- The test cases are difficult to design.

Regression Testing

- Regression testing is **testing existing software applications to make sure that a change or addition hasn't broken any existing functionality**.

Example, these code changes could include adding new features, fixing bugs, or updating a current feature.



Acceptance Testing

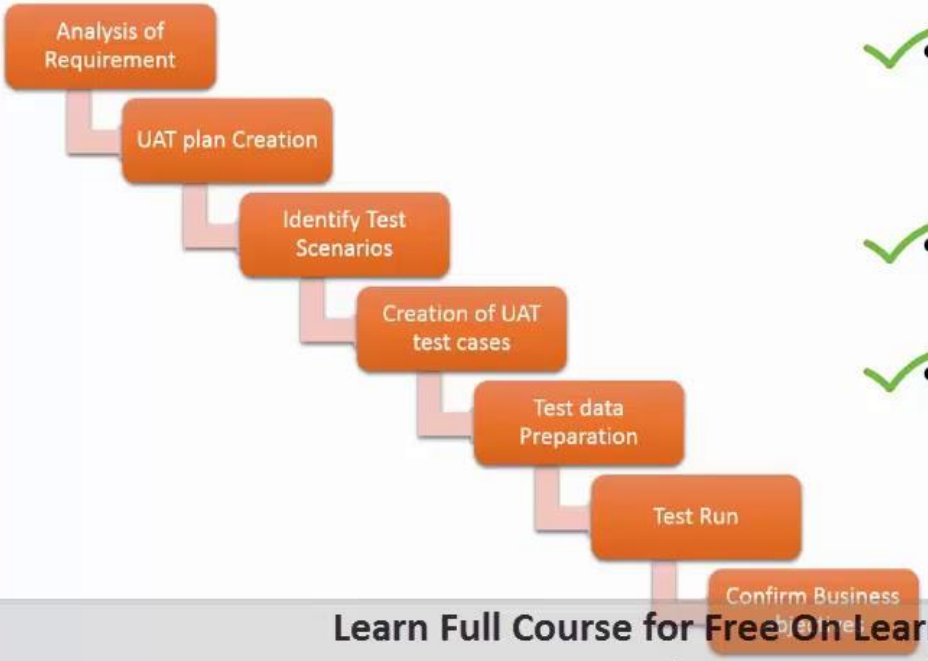
- Acceptance testing is **a quality assurance (QA) process that determines to what degree an application meets end users' approval**. Depending on the organization, acceptance testing might take the form of beta testing, application testing, field testing or end-user testing.

Example:

Alpha and beta testing are examples of acceptance testing. Alpha tests are internal and aim to spot any glaring defects, while beta testing is an external pilot-test of a product before it goes into commercial production.

Acceptance Testing

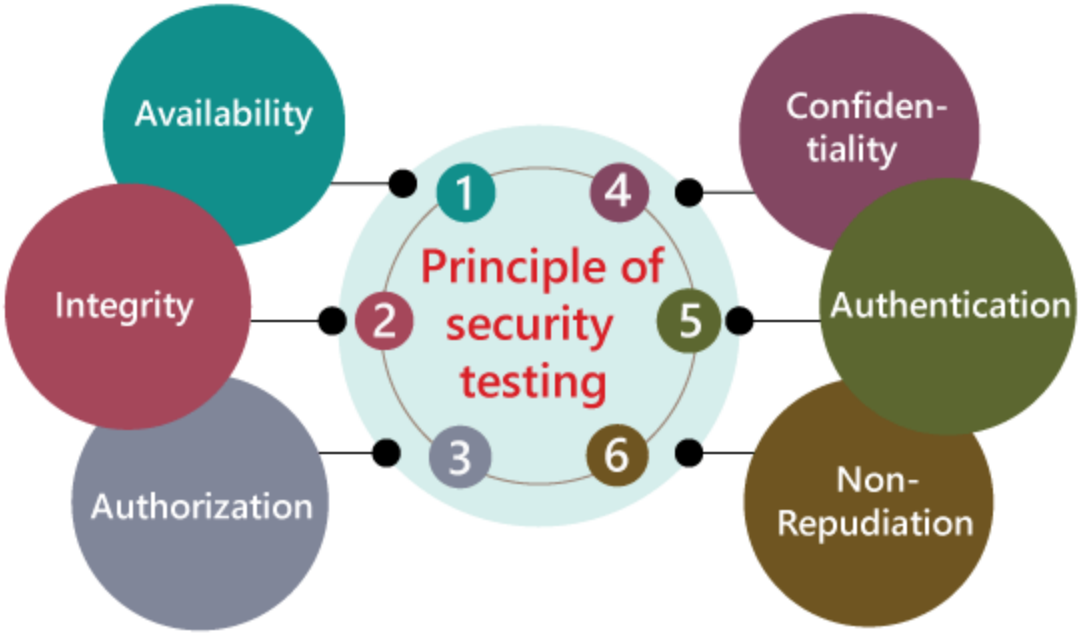
User AcceptanceTesting Process



- No critical defects in software testing open.
- Business process works satisfactorily.
- UAT Sign off meeting with all stakeholders.

Security Testing

- Software security testing is a software testing process that ensures the software is free of any potential vulnerabilities or weaknesses, risks, or threats so that the software might not harm the user system and data.



Security Testing

6 Principle of Security testing

1. **Availability:-**In this, the data must be retained by an official person, and they also guarantee that the data and statement services will be ready to use whenever we need it.

2. **Integrity:-**In this, we will secure those data which have been changed by the unofficial person. The primary objective of integrity is to permit the receiver to control the data that is given by the system

3. **Authorization:-**

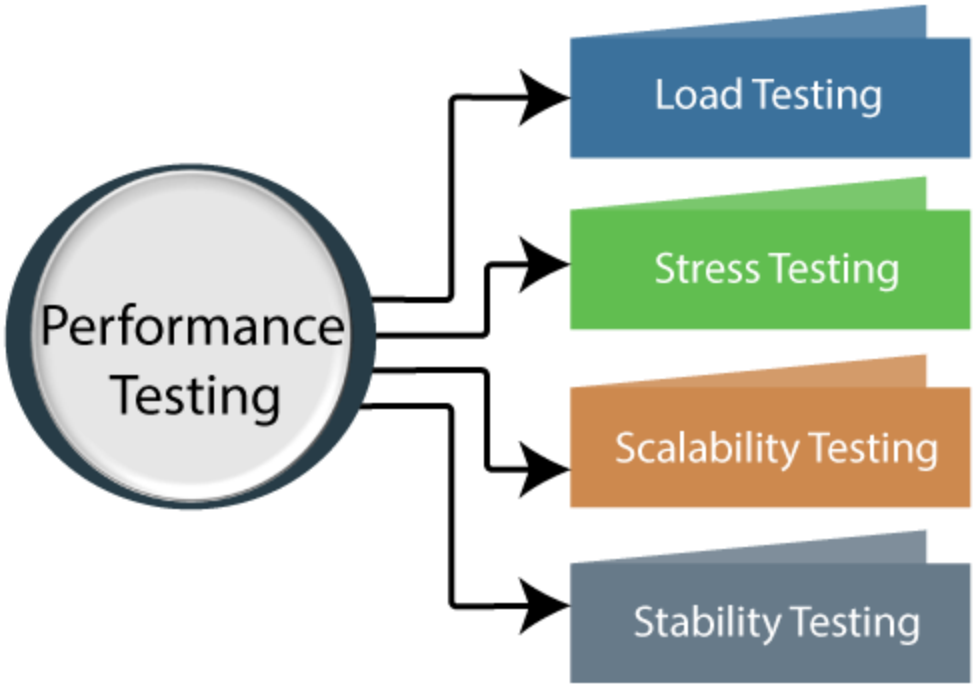
4. **Confidentiality**:-It is a security process that protracts the leak of the data from the outsider's

5. **Authentication**:-The authentication process comprises confirming the individuality of a person

6. **Non-repudiation:-**The non-repudiation is used to ensure that a conveyed message has been sent and received by the person who claims to have sent and received the message.

Performance Testing

- Performance testing is **a non-functional software testing technique that determines how the stability, speed, scalability, and responsiveness of an application holds up under a given workload**



Performance Testing

Benefits of Performance Testing

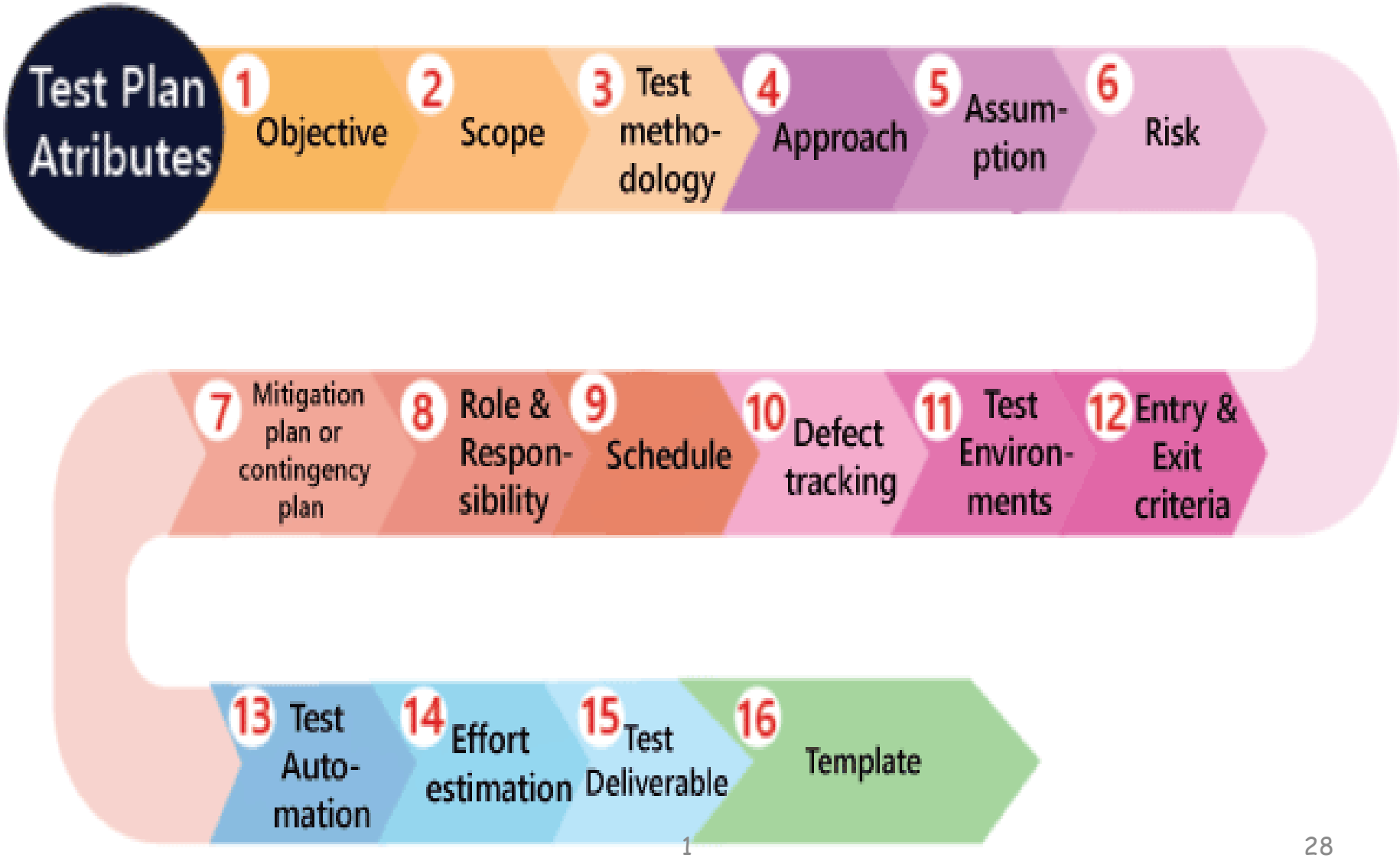
- Validate the fundamental features of the software.
- Measure the speed, accuracy and stability of software.
- Performance testing allows you to keep your users happy.
- Identify discrepancies and resolve issues.
- Improve optimization and load capability.

What is Test Planning?

- Define the functions, roles and methods for all test phases.
- Test planning usually start during the requirements phase.
- Major test plan elements are:

1. Objectives for each test phase
2. Schedules and responsibilities for each test activity
3. Availability of tools, facilities and test libraries.
4. Set the criteria for test completion

Test Planning



Test Execution & Reporting

- Testing should be treated like an experiment.
- Testing require that all anomalous behavior be noted and investigated.
- Big companies keep a special library with all copies of test reports, incident forms, and test plans

Test Execution & Reporting

Test Report

Test Cycle System Test

EXECUTED	PASSED			130	
	FAILED			0	
	(Total) TESTS EXECUTED (PASSED + FAILED)				**130**
PENDING					0
IN PROGRESS					0
BLOCKED					0
(Sub-Total) TEST PLANNED (PENDING + IN PROGRESS + BLOCKED + TEST EXECUTED)					**130**

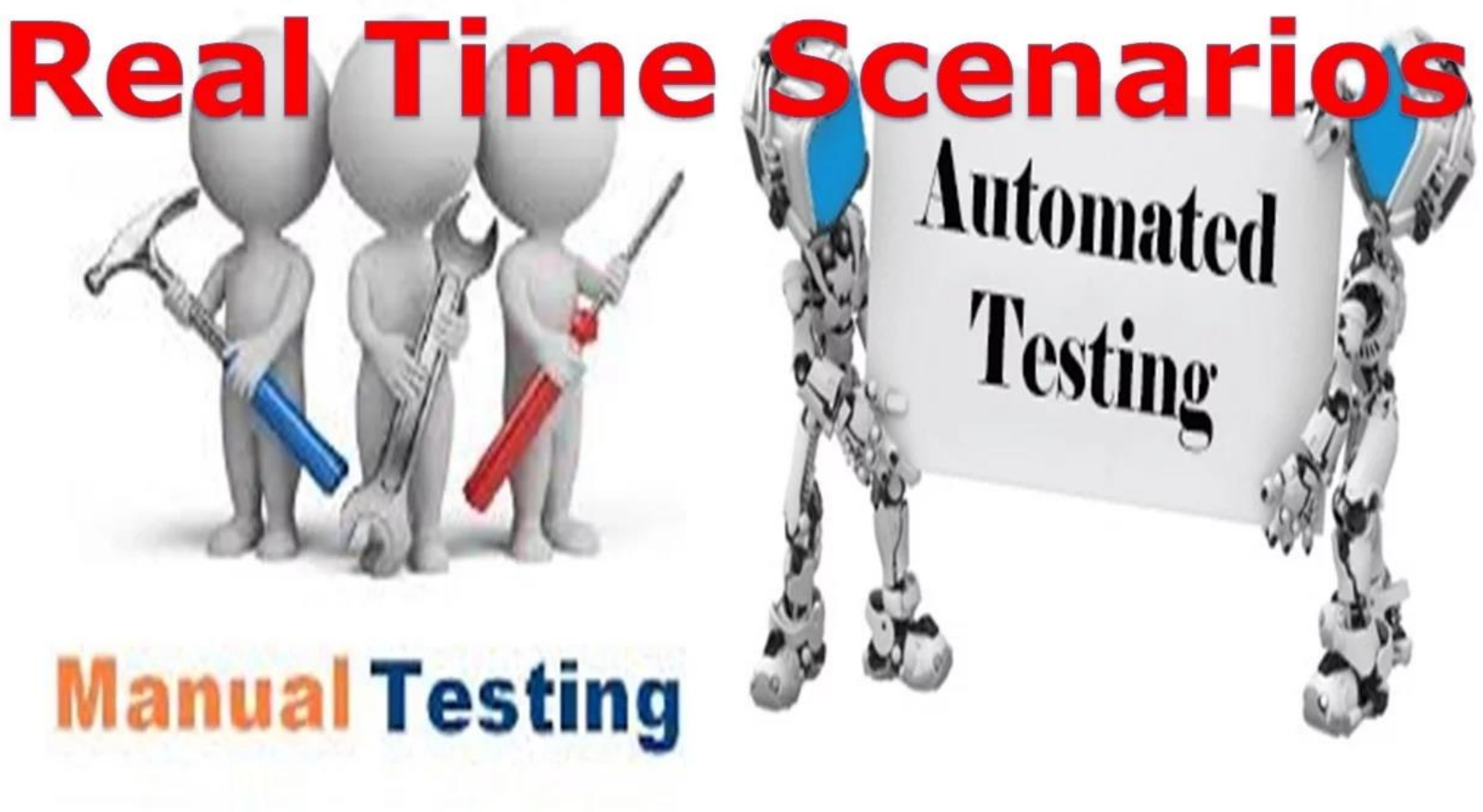
Functions	Description	% TCs Executed	% TCs Passed	TCs pending	Priority	Remarks
New Customer	Check new Customer is created	100%	100%	0	*High*	
Edit Customer	Check Customer can be edited	100%	100%	0	*High*	
New Account	Check New account is added	100%	100%	0	*High*	
Edit Account	Check Account is edit	100%	100%	0	*High*	
Delete Account	Verify Account is delete	100%	100%	0	*High*	
Delete customer	Verify Customer is Deleted	100%	100%	0	*High*	
Mini Statement	Verify Ministatement is generated	100%	100%	0	*High*	
Customized Statement	Check Customized Statement is generated	100%	100%	0	*High*	

Real-Time Testing

- Real-Time testing is necessary because the deployment system is usually more complicate than development system
- Rules apply for testing real time system

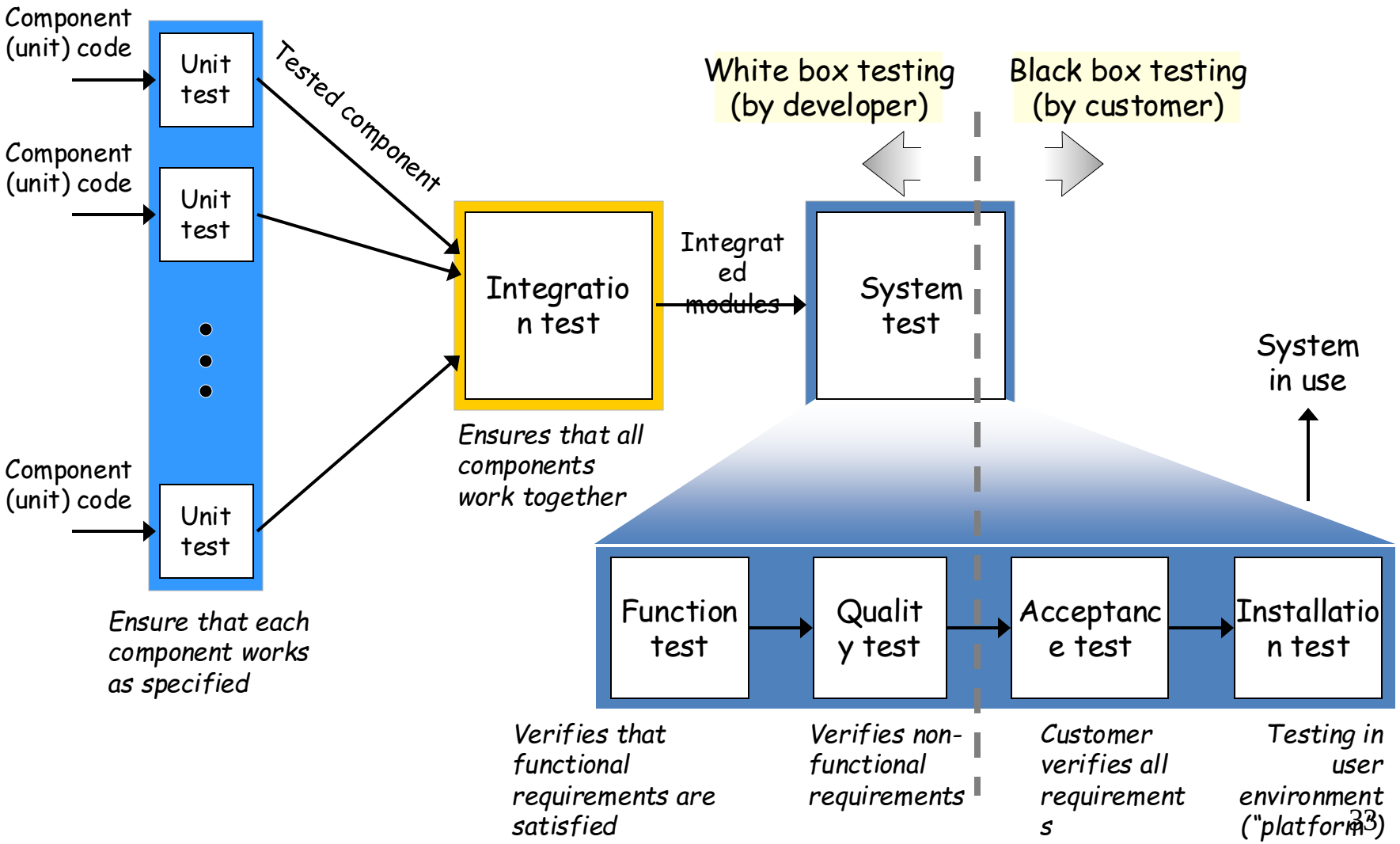
1. Evaluate possible deadlocks, thrashing to special timing conditions
2. Use tests to simulate hardware faults.
3. Use hardware simulation to stress the software design.
4. Design ways to simulate modules missing in the development system.

Real-Time Testing



Logical Organization of Testing

(Usually not done in a linear step-by-step order and completed when the last step is reached!)



Thank You