Section-C

Lecture-23

Dronacharya College of Engineering

Introduction to Software Testing

What is Software Testing

Several definitions:

"Testing is the process of establishing confidence that a program or system does what it is supposed to." by Hetzel 1973

"Testing is the process of executing a program or system with the intent of finding errors." by Myers 1979

"Testing is any activity aimed at evaluating an attribute or capability of a program or system and determining that it meets its required results." by Hetzel 1983

Software Testing

Software testing is a process used to identify the correctness, completeness and quality of developed computer software.

It is the process of executing a program / application under positive and negative conditions by manual or automated means. It checks for the :-

- Specification
- Functionality
- Performance

Testing Objectives

The Major Objectives of Software Testing:

- Uncover as many as errors (or bugs) as possible in a given timeline.

- Demonstrate a given software product matching its requirement specifications.

- Validate the quality of a software testing using the minimum cost and efforts.

- Generate high quality test cases, perform effective tests, and issue correct and helpful problem reports.

Major goals:

uncover the errors (defects) in the software, including errors in:

- **requirements from requirement analysis**
- **design documented in design specifications**
- **coding (implementation)**
- **system resources and system environment**
- **hardware problems and their interfaces to software**

What ….????

…is an ”ERROR”??

….is a ”Bug”??

….is Fault, Failure ??

Bug, Fault & Failure

A person makes an **Error**
That creates a **fault** in software
That can cause a **failure** in operation

Error : **An error is a human action that produces the incorrect result that results in a fault.**

Bug : **The presence of error at the time of execution of the software.**

Fault : **State of software caused by an error.**

Failure : **Deviation of the software from its expected result. It is an event.**

Who does Software Testing

- Test manager
  - manage and control a software test project
  - supervise test engineers
  - define and specify a test plan

- Software Test Engineers and Testers
  - define test cases, write test specifications, run tests

- Independent Test Group

- Development Engineers
  - Only perform unit tests and integration tests

- Quality Assurance Group and Engineers
  - Perform system testing
  - Define software testing standards and quality control process

Who is a Software Tester??..

Software Tester is the one who performs testing and find bugs, if they exist in the tested application.

Software Testing Activities

- **Test Planing**

  Define a software test plan by specifying:

  - **a test schedule for a test process and its activities, as well as assignments**
  - **test requirements and items**
  - **test strategy and supporting tools**

- **Test Design and Specification**

  - **Conduct software design based well-defined test generation methods.**
  - **Specify test cases to achieve a targeted test coverage.**

- **Test Set up:**

  - **Testing Tools and Environment Set-up**

- **Test Operation and Execution**

  - **Run test cases manually or automatically**

Software Testing Activities

- Test Result Analysis and Reporting

Report software testing results and conduct test result analysis

- Problem Reporting

Report program errors using a systematic solution.

- Test Management and Measurement

Manage software testing activities, control testing schedule, measure testing complexity and cost

- Test Automation

- Define and develop software test tools
- Adopt and use software test tools
- Write software test scripts and facility

Software Testability

Software testability means how easily a computer program can be tested. There are certain metrics that can be used to measure testability
Following are some key characteristics of testability.

1. **Operability:** ***the better it works, the more efficient is testing process***

2. **Observability:** ***what you see is what you test***

3. **Controllability:** ***the better it is controlled , the more we can automate the testing process***

4. **Decomposability:** ***by controlling the scope of testing, we can more quickly isolate problems and perform smarter testing.***

5. **Simplicity:** ***the less there is to test, the more quickly we can test it.***

6. **Stability:** ***the fewer the changes .***

7. **Understandability:** ***the more information we have ,the smarter we will test***

Software Testing Principles

- Principle #1: Complete testing is impossible.

- Principle #2: Software testing is not simple.
  - Reasons:
    - Quality testing requires testers to understand a system/product completely
    - Quality testing needs adequate test set, and efficient testing methods
    - A very tight schedule and lack of test tools.

- Principle #3: Testing is risk-based.

- Principle #4: Testing must be planned.

- Principle #5: Testing requires independence.

- Principle #6: Quality software testing depends on:
    - Good understanding of software products and related domain application
    - Cost-effective testing methodology, coverage, test methods, and tools.
    - Good engineers with creativity, and solid software testing experience

Software Testing Principles

- Principle #7: All test should be based on customer requirements
- Principle #8: Software testing should be planned long before testing begins
- Principle #9: Document test cases and test results.
- Principle #10: Use effective resources to test.

Software Testing Myths

- We can test a program completely. In other words, we test a program exhaustively.

- We can find all program errors as long as test engineers do a good job.

- We can test a program by trying all possible inputs and states of a program.

- A good test suite must include a great number of test cases.

- Good test cases always are complicated ones.

- Software test automation can replace test engineers to perform good software testing.

- Software testing is simple and easy. Anyone can do it. No training is needed.

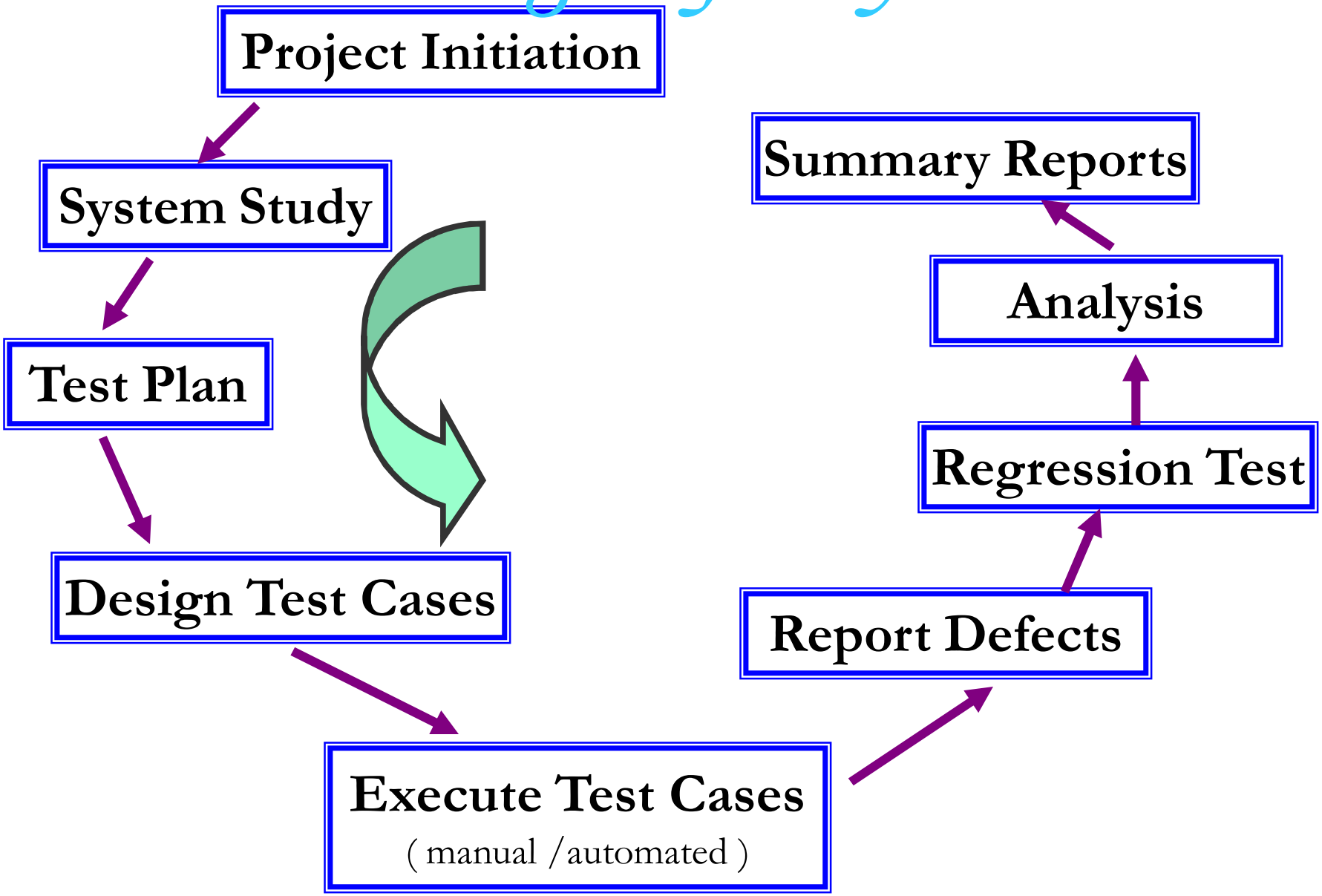
Importance Of Testing In SDLC

When to Start Testing in SDLC

- **Requirement**
- **Analysis**
- **Design**
- **Coding**
- **Testing**
- **Implementation**
- **Maintenance**

❖ ***Testing starts from Requirement Phase***

Testing Life Cycle



Test Planning

Test Plan

A test plan is a systematic approach to testing a system i.e. software. The plan typically contains a detailed understanding of what the eventual testing workflow will be.

Test Case

A test case is a specific procedure of testing a particular requirement.

OR

A test case has components that describe an input, action or event and an expected response, to determine if a feature of an application is working correctly

Why we write Test Cases ?

The basic objective of writing test cases is to validate the testing coverage of the application. If you are working in any CMM company then you will strictly follow test cases standards.

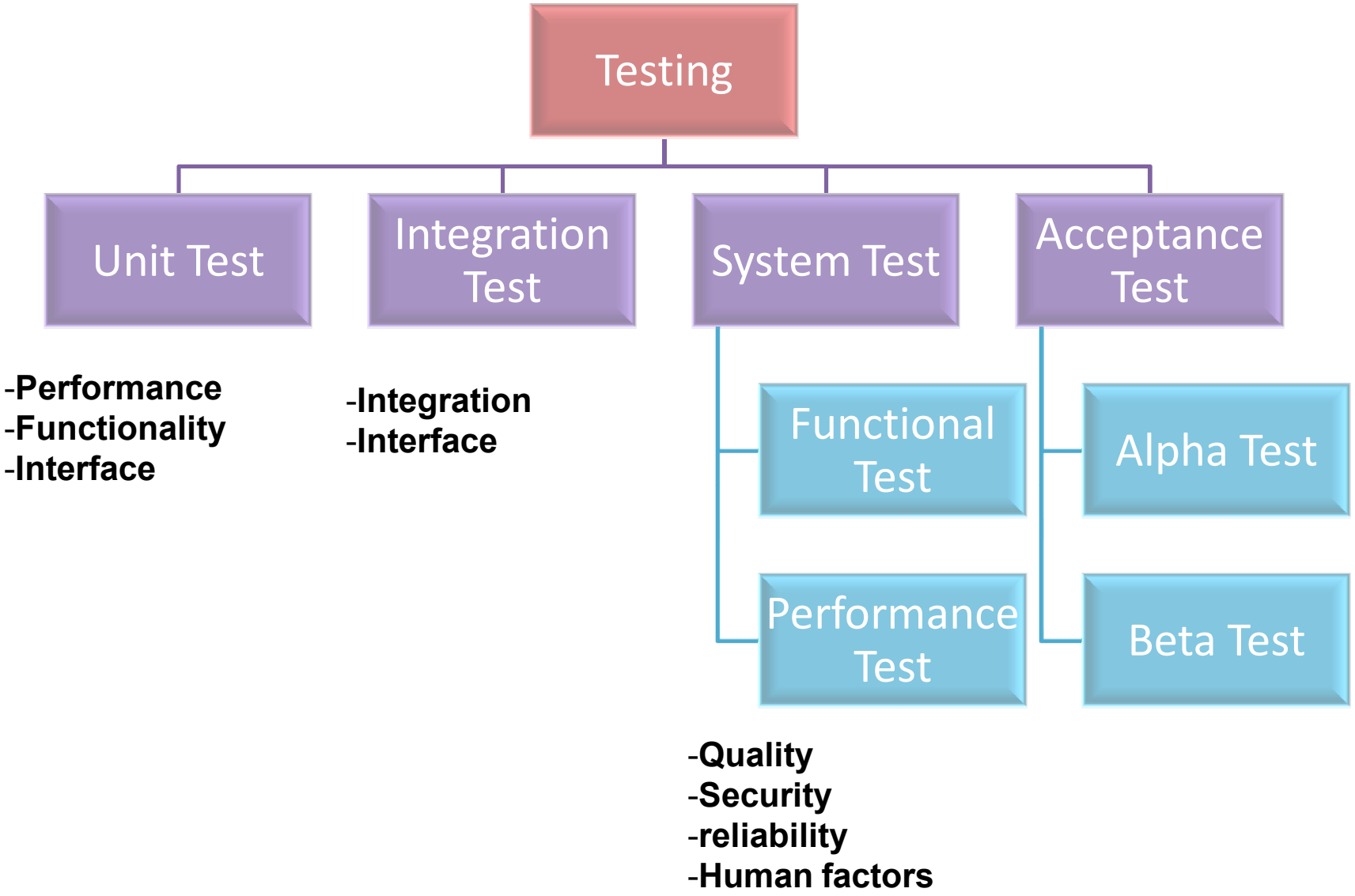
How to write a test cases?

There is a format to write a test case.

- Field in test cases:
- Test case id:
- Unit to test: ***what to be verified***
- Assumptions
- Test Data: ***variables and their values***
- Steps to be executed:
- Expected results:
- Actual result:
- Pass/Fail:
- Comments:

Levels Of Testing

Levels Of Testing



Unit Test

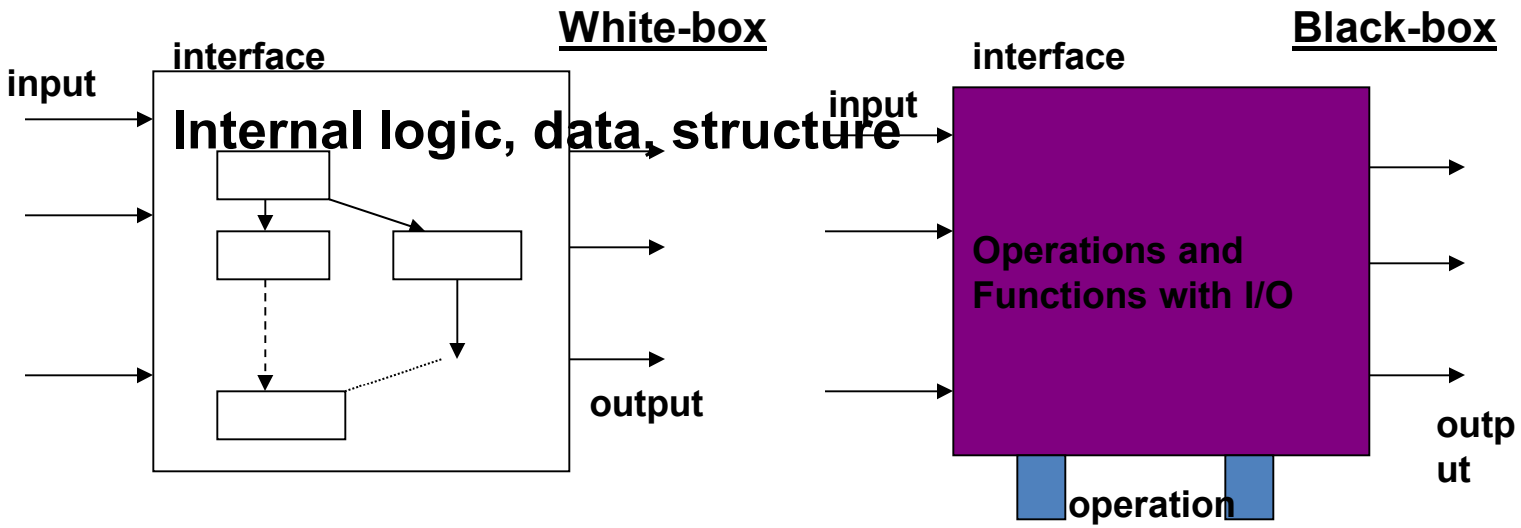
1. The firs level in the testing process is called unit testing.
2. Unit testing concerns testing smallest component of the software
3. Unit testing is done by Developer.

Unit Test (Component Level Test)

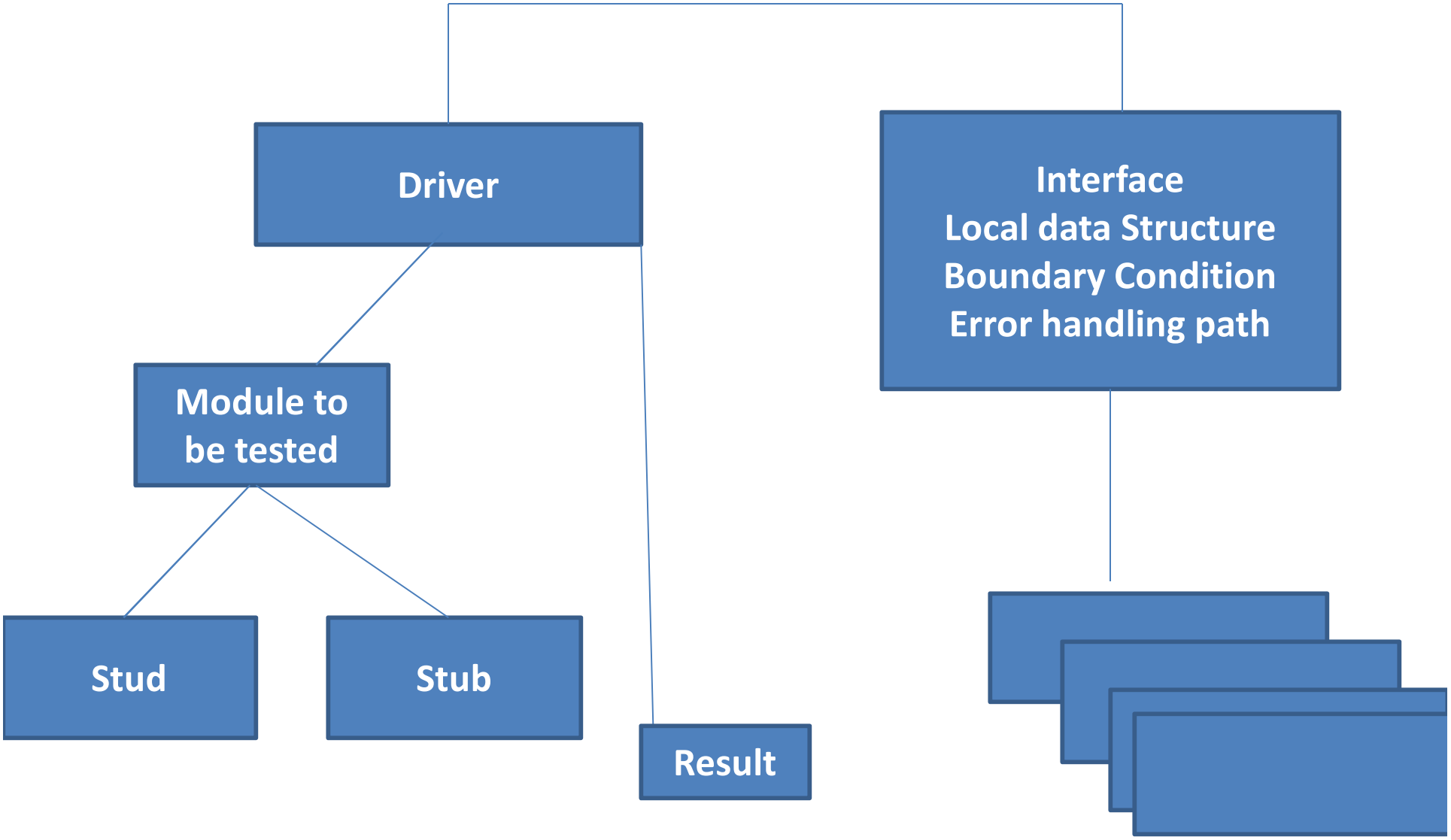
Unit testing: Individual components are tested independently to ensure their quality. The focus is to uncover errors in design and implementation, including

- data structure in a component
- program logic and program structure in a component
- component interface
- functions and operations of a component

Unit testers: developers of the components.



Unit Test Procedures



Unit Test Procedures

1. Each test case should be linked with a set of anticipated results.
2. As a module is not a stand alone program, driver and stub software must be produced for each test units.
3. In most of applications a *driver* is nothing than a "main program" that accept test case data, passes such data to the component(to be tested), and print relevant results.
4. *Stubs* serve to replace modules that are subordinate to the component to be tested

Integration Testing

After completing the unit testing and dependent modules development, programmers connect the modules with respect to HLD for Integration Testing through below approaches.

A study has shown that almost 40% of the error are due to integration and interface problems. there are number of strategies that can be followed to do integration testing.

- Incremental Strategy
- Non- Incremental Strategy
- Mixed Strategy

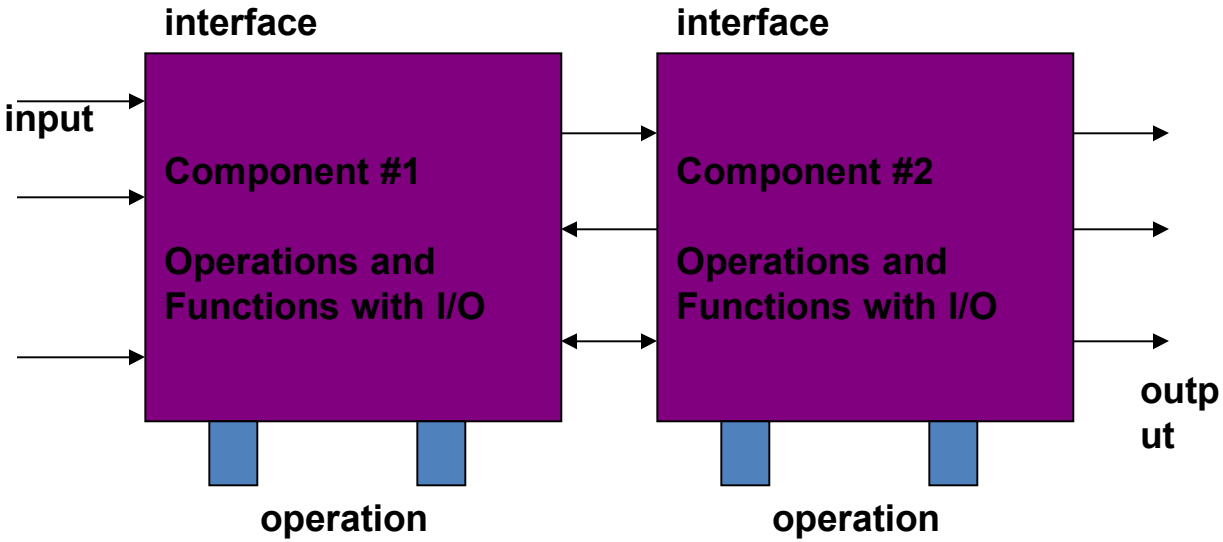
Integration Testing

Integration test: A group of dependent components are tested together to ensure their the quality of their integration unit.

The focus is to uncover errors in:

- Design and construction of software architecture
- Integrated functions or operations at sub-system level
- Interfaces and interactions between them
- Resource integration and/or environment integration

Integration testers: either developers and/or test engineers.



Approaches to Integration Testing

The various approaches used for integration testing are:

- Big Bang Approach
- Incremental Approach
- Top Down Integration Testing
- Bottom-up Integration Testing
- Sandwich Integration Testing
- Regression Testing

System Testing

After completing **Unit** and **Integration** testing through white box testing techniques development team release an .exe build (all integrated module) to perform black box testing.

- Usability Testing
- Functional Testing
- Performance Testing
- Security Testing

System Testing

System test: The system software is tested as a whole. It verifies all elements mesh properly to make sure that all system functions and performance are achieved in the target environment.

The focus areas are:

- System functions and performance
- System reliability and recoverability (recovery test)
- System installation (installation test)
- System behavior in the special conditions (stress and load test)
- System user operations (acceptance test/alpha test)
- Hardware and software integration and collaboration
- Integration of external software and the system

System testers: test engineers in ITG or SQA people.

When a system is to be marketed as a software product, a testing process called beta testing is often used.

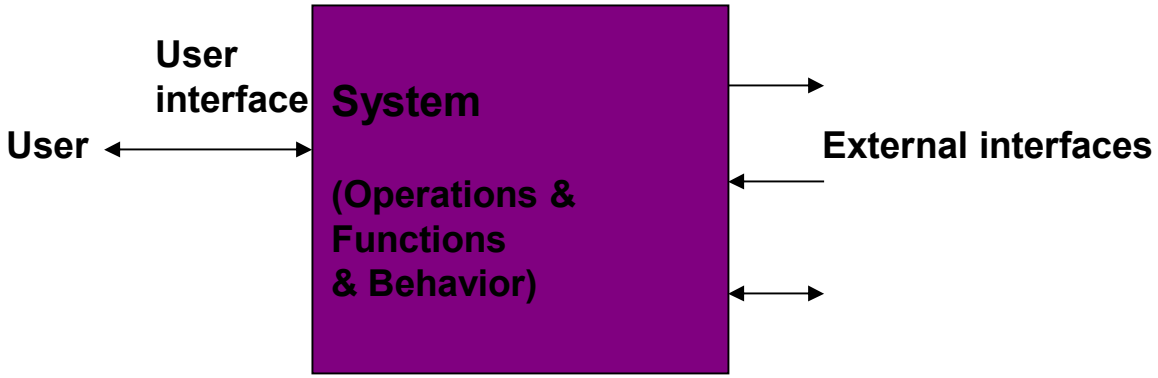
Function Validation Testing

Validation test: The integrated software is tested based on requirements to ensure that we have a right product.

The focus is to uncover errors in:

- System input/output
- System functions and information data
- System interfaces with external parts
- User interfaces
- System behavior and performance

Validation testers: test engineers in ITG or SQA people.



Usability Testing

During this test, testing team concentrates on the user friendliness of build interface. It consists of following sub tests.

- **User Interface Test:** Ease of use (screens should be understandable to operate by End User)
- **Look & Feel :-** attractive
- **Speed in interface :-** Less number of task to complete task
- **Manual Support Test :-** Context sensitiveness of user manual.

Functional Testing

- The process of checking the behavior of the application.
- It is geared to functional requirements of an application.
- To check the correctness of outputs.
- Data validation and Integration i.e. inputs are correct or not.

Performance Testing

- **LOAD TESTING –** Also Known as Scalability Testing. During this test, test engineers execute application build under customer expected configuration and load to estimate performance.

- **STRESS TESTING –** During this test, Test engineers estimates the peak load. To find out the maximum number of users for execution of out application user customer expected configuration to estimate peak load.

  PEAK LOAD > CUSTOMER LAOD (EXPECTED)

- **DATA VOLUME TESING --** Testing team conducts this test to find the maximum limit of data volume of your application.

Security Testing

Testing how well the system protects against unauthorized internal or external access, willful damage, etc, may require sophisticated testing techniques

Smoke testing

Smoke testing is non-exhaustive software testing, ascertaining that the most crucial functions of a program work, but not bothering with finer details.

Alpha Testing

1. The application is tested by the users who doesn't know about the application.
2. Done at developer's site under controlled conditions
3. Under the supervision of the developers.

Acceptance Testing

A formal test conducted to determine whether or not a system satisfies its acceptance criteria and to enable the customer to determine whether or not to accept the system.

It is the final test action before deploying the software. The goal of acceptance testing is to verify that the software is ready and can be used by the end user to perform the functions for which the software was built.

Beta Testing

1. This Testing is done before the final release of the software to end-users.

1. Before the final release of the software is released to users for testing where there will be no controlled conditions and the user here is free enough to do what ever he wants to do on the system to find errors.

Regression Testing

Testing with the intent of determining if bug fixes have been successful and have not created any new problems. Also, this type of testing is done to ensure that no degradation of baseline functionality has occurred.

Monkey Testing

Testing the application randomly like hitting keys irregularly and try to breakdown the system there is no specific test cases and scenarios for monkey testing.

Verification
&
Validation

Verification and Validation

Software testing is one element of a broader topic that is often referred to as
===> Verification and Validation (V&V)

Verification --> refers to the set of activities that ensure that software correctly implements a specific function.

Validation -> refers to a different set of activities that ensure that the software that has been built is traceable to customer requirements.

Boehm [BOE81]:

Verification: "Are we building the product right?"
Validation: "Are we building the right product?"

The definition of V&V encompasses many of SQA activities, including
formal technical reviews, quality and configuration audits
performance monitoring, different types of software testing
feasibility study and simulation

Verification

- **Verification is the process confirming that -software meets its specification, done through inspections and walkthroughs**

***Use* – To identify defects in the product early in the life cycle**

Validation

- **Validation is the process confirming that it meets the user's requirements. It is the actual testing.**

Verification : Is the Product Right

Validation : Is it the Right Product

Test Issues in Real World

Software testing is very expensive.

How to achieve test automation?????

When should we stop software testing?

Test criteria, test coverage, adequate testing.

Other software testing:

- **GUI Testing**
- **Object-Oriented Software Testing**
- **Component Testing and Component-based Software Testing**
- **Domain-specific Feature Testing**
- **Testing Web-based Systems**

Quality Assurance

&

Quality Control

What is Quality ?

Quality is defined as meeting the customer's requirements and according to the standards

The best measure of Quality is given by FURPS

- **Functionality**
- **Usability**
- **Reliability**
- **Performance**
- **Scalability**

Why Quality ?

- **Quality is the important factor affecting an organization's long term performance.**

- **Quality improves productivity and competitiveness in any organization.**

Quality Assurance

Quality Assurance is a planned and systematic set of activities necessary to provide adequate confidence that products and services will conform to specified requirements and meets user needs.

- It is process oriented.
- Defect prevention based.
- Throughout the Life Cycle.
- It's a management process.

Quality Control

Quality control is the process by which product quality is compared with the applicable standards and the action taken when non conformance is detected.

- It is product oriented
- Defect detection based

QA vs. QC

- Quality Assurance makes sure that we are doing the right things, the right Way.
- QA focuses on building in quality and hence preventing defects.
- QA deals with process.
- QA is for entire life cycle.
- QA is preventive process.

- Quality Control makes sure the results of what we've done are what we expected .
- QC focuses on testing for quality and hence detecting defects.
- QC deals with product.
- QC is for testing part in SDLC.
- QC is corrective process.

Bug Reporting

Bug Reporting Format

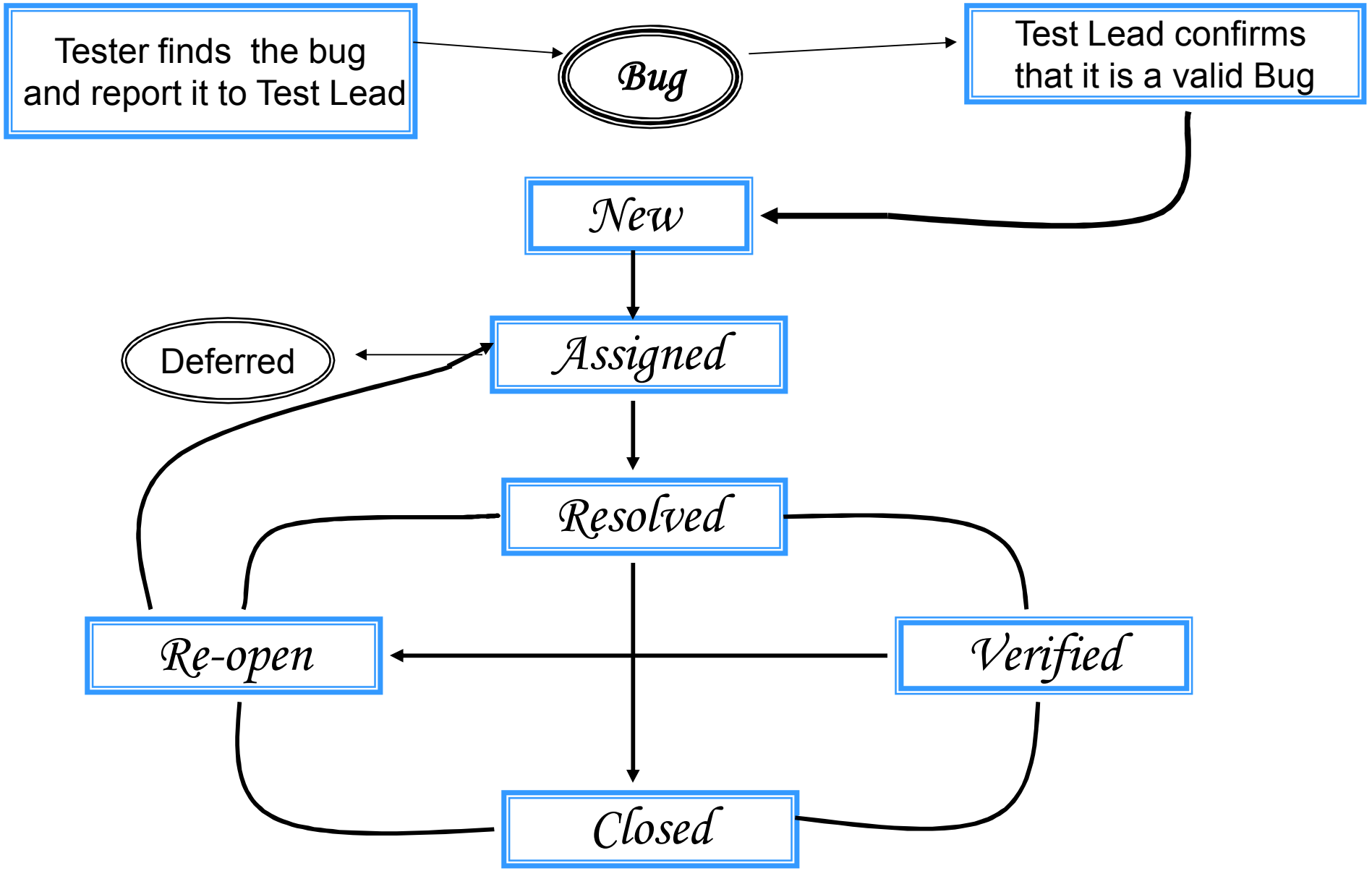
Bug Id	**:**	**Sys_def_xyz_o1**
Test case Id	**:**	**Sys_xyz_o1**
Bug Description	**:**	**Specify the defect.**
Bug Severity	**:**	**Severity of the Defect**
Bug Priority	**:**	**Major or Minor**
Bug Status	**:**	**Status of the Defect**
Identified by	**:**	**ABC**
Module	**:**	**Database Operations**
Project	**:**	**XYZ Corporation**
Dated	**:**	**22/9/03**

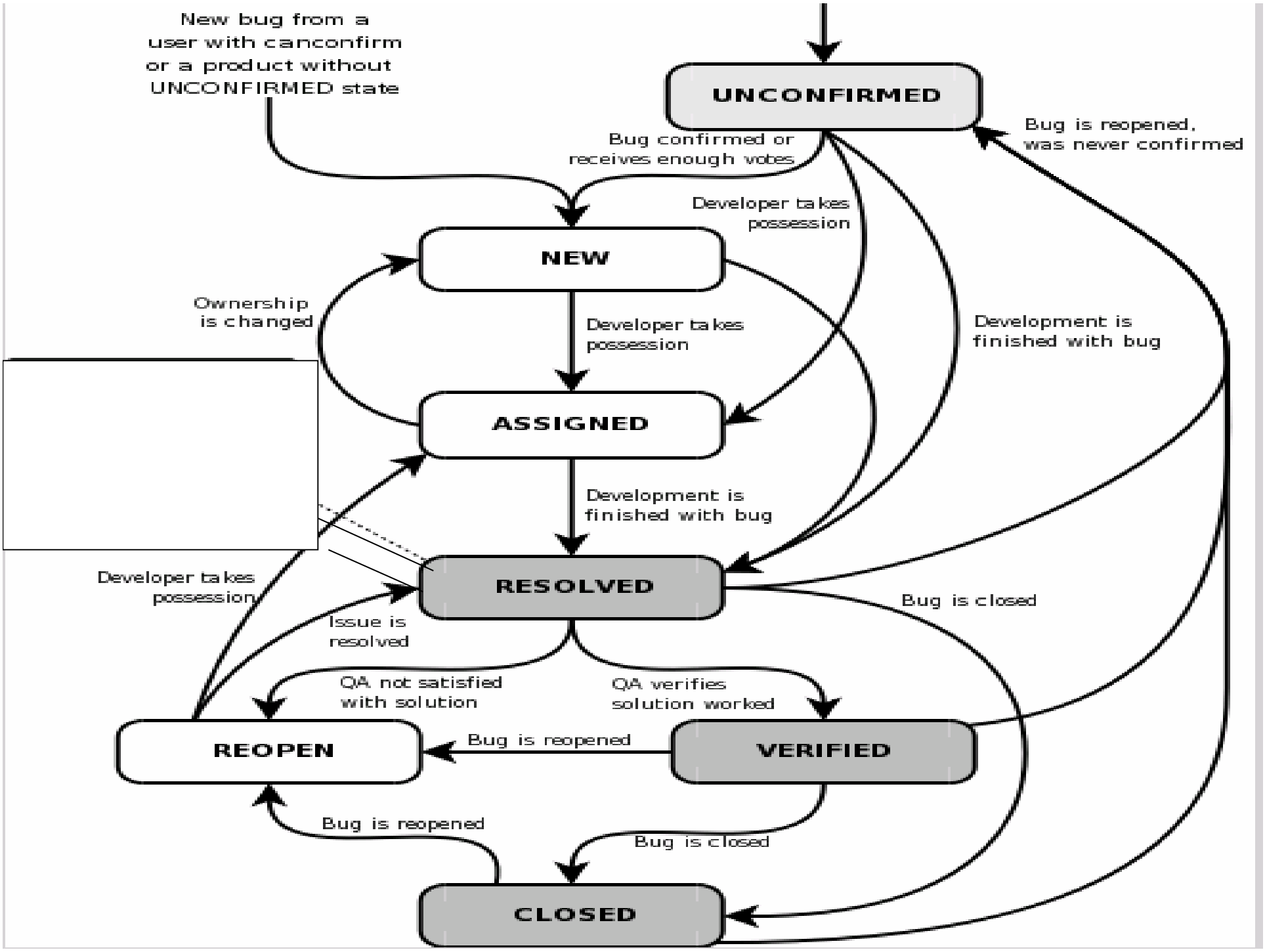
Bug Priority

Priority- How important it is to correct that specific bug

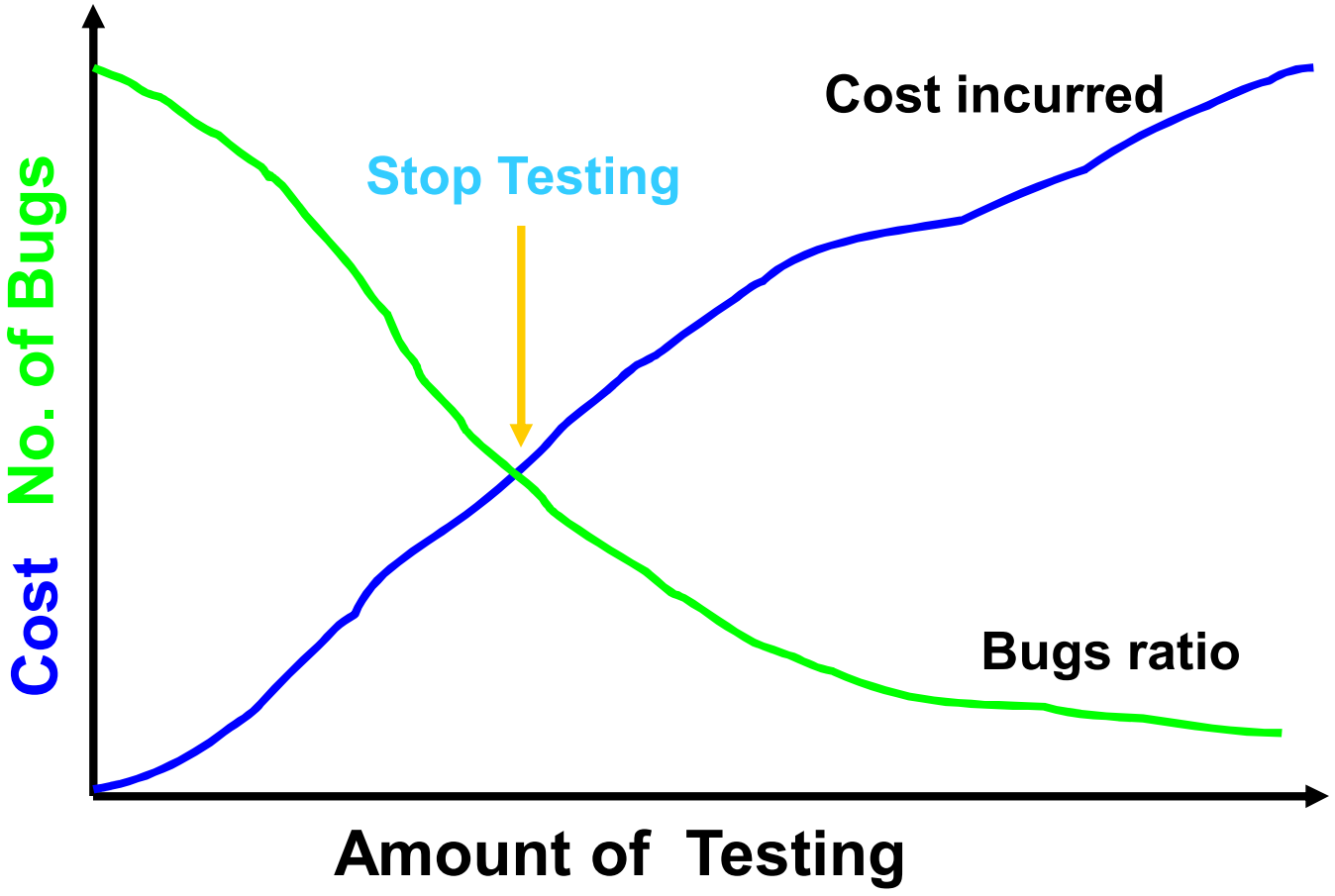
- High
- Moderate
- Low

Bug Life Cycle




New bug from a user with canconfirm or a product without UNCONFIRMED state
UNCONFIRMED
Bug confirmed or receives enough votes
Bug is reopened, was never confirmed
Developer takes possession
NEW
Ownership is changed
Developer takes possession
Development is finished with bug
ASSIGNED
Development is finished with bug
Developer takes possession
RESOLVED
Bug is closed
Issue is resolved
QA not satisfied with solution
QA verifies solution worked
REOPEN
Bug is reopened
VERIFIED
Bug is reopened
Bug is closed
CLOSED

When to Stop Testing



Software Testing Limits

- Due to the testing time limit, it is impossible to achieve total confidence.

- We can never be sure the specifications are 100% correct.

- We can never be certain that a testing system (or tool) is correct.

- No testing tools can copy with every software program.

- Tester engineers never be sure that they completely understand a software product.

- We never have enough resources to perform software testing.

- We can never be certain that we achieve 100% adequate software testing.