Introduction to Software Testing

A foundational lecture on software quality assurance.

Saeed Siddik
Assistant Professor
IIT University of Dhaka

What is Software Testing?

- It is the process of evaluating and verifying that a software product or application does what it is supposed to do.
- The primary objectives are:
  - To find defects, errors, or gaps in the software.
  - To ensure the product meets all specified requirements.
  - To guarantee a high level of quality, reliability, and performance.
  - To reduce the risk of failure in the production environment.

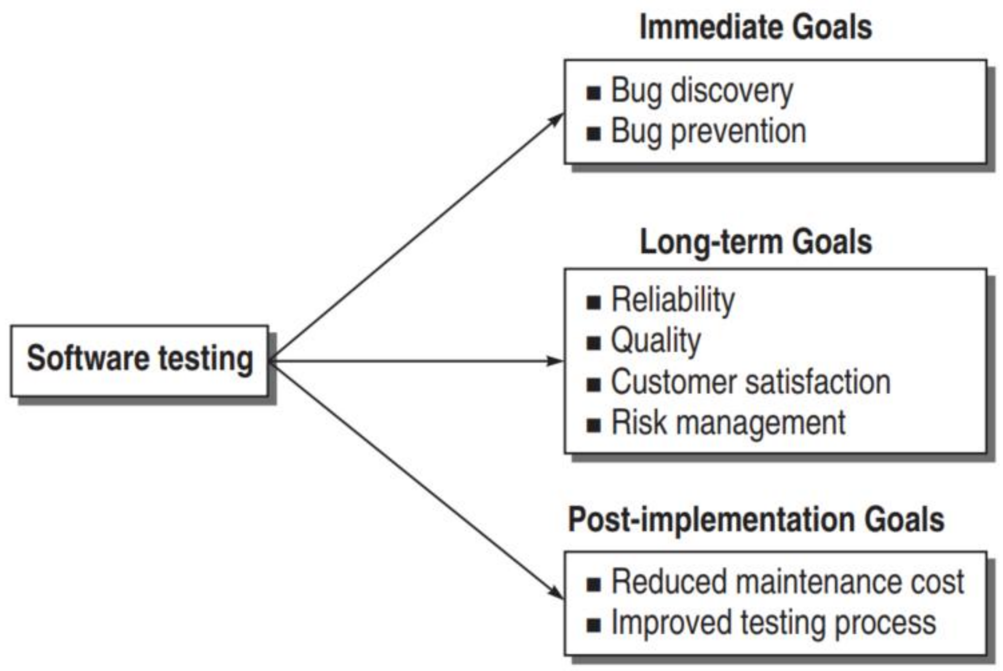
The evolution of software testing

- **Software Testing 1.0**
  - Considered as a single phase performed after coding in SDLC.
  - No test organization was there
- **Software Testing 2.0**
  - Early testing started in SDLC .
  - Testing was evolving in the direction of planning the test resources.
- **Software Testing 3.0**
  - Evolved in the form of a process which is based on strategic effort.
  - Prepare a roadmap of the overall testing process before coding.
  - Driven by quality goals so that all controlling and monitoring activities can be performed by the managers.

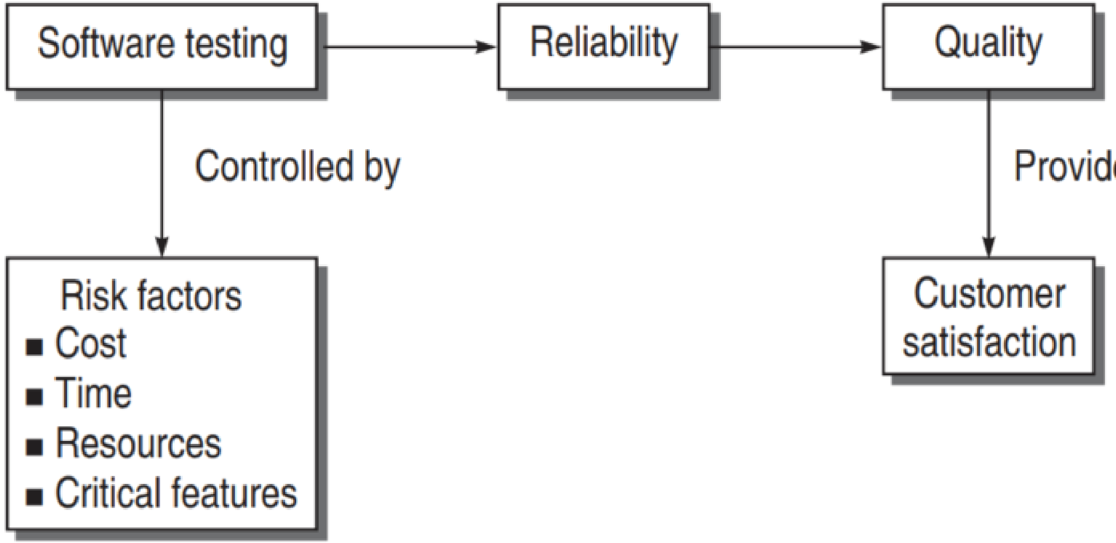
Software testing— myths and facts

- Myth Testing is a single phase in SDLC
- Myth Testing is easy.
- Myth Software development is worth more than testing
- Myth Complete testing is possible.
- Myth Testing starts after program development
- Myth Anyone can be a tester.

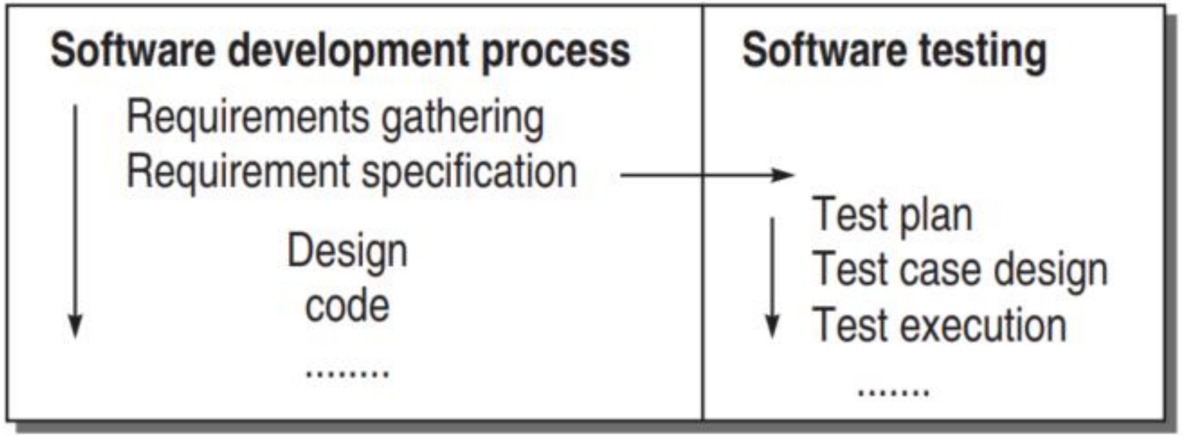
Goals of software testing



Testing controlled by risk factors

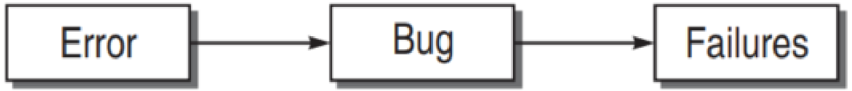


Testing process runs parallel to software process



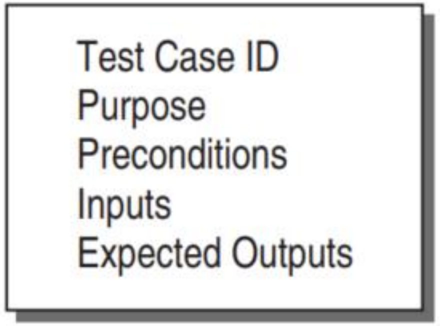
Fault/Defect/Bug vs Error

- *Fault* is a condition that in actual causes a system to produce failure
- Fault is synonymous with the words *defect* or *bug*.
- One failure may be due to one or more bugs and one bug may cause one or more failures.
- Whenever a development team member makes a mistake in any phase of SDLC, errors are produced.
- An error causes a bug and the bug in turn causes failures



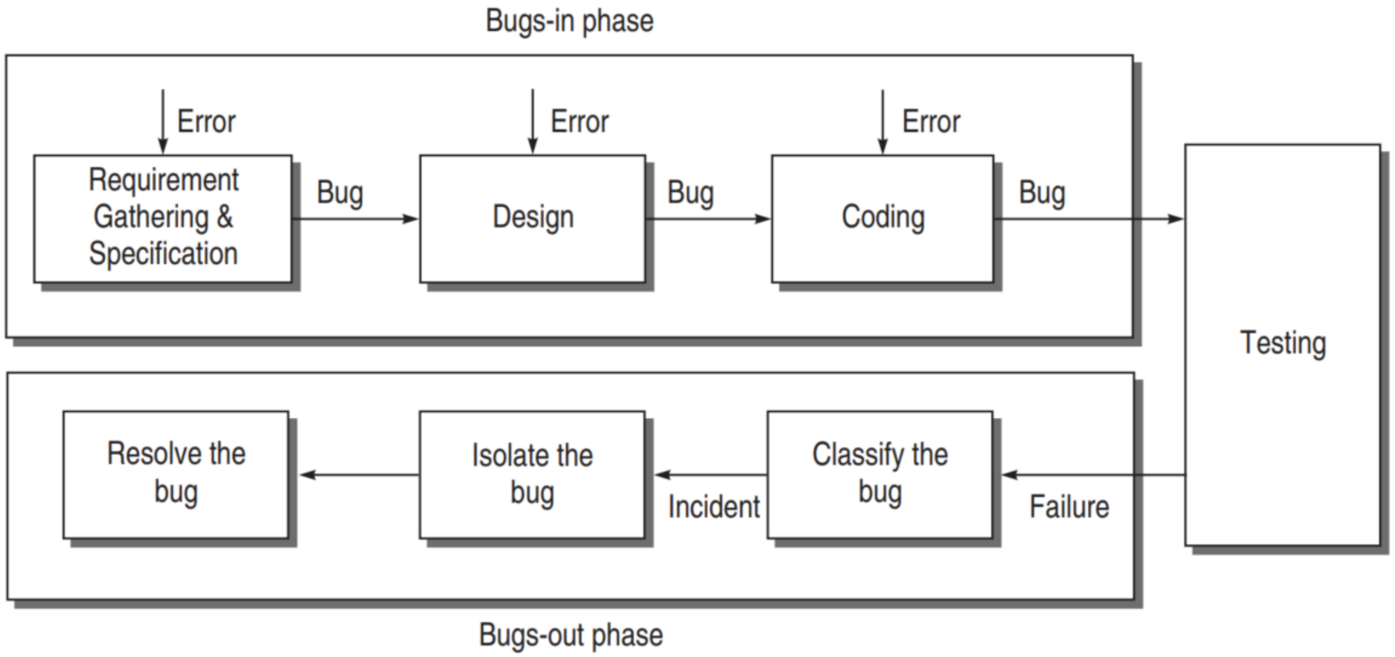
Test case template

- *Test case ID* is the identification number given to each test case.
- *Purpose* defines why the case is being designed.
- *Preconditions* for running the inputs in a system can be defined, if required, in a test case
- *Inputs* should not be hypothetical. Actual inputs must be provided, instead of general inputs
- *Expected outputs* are the outputs which should be produced when there is no failure.

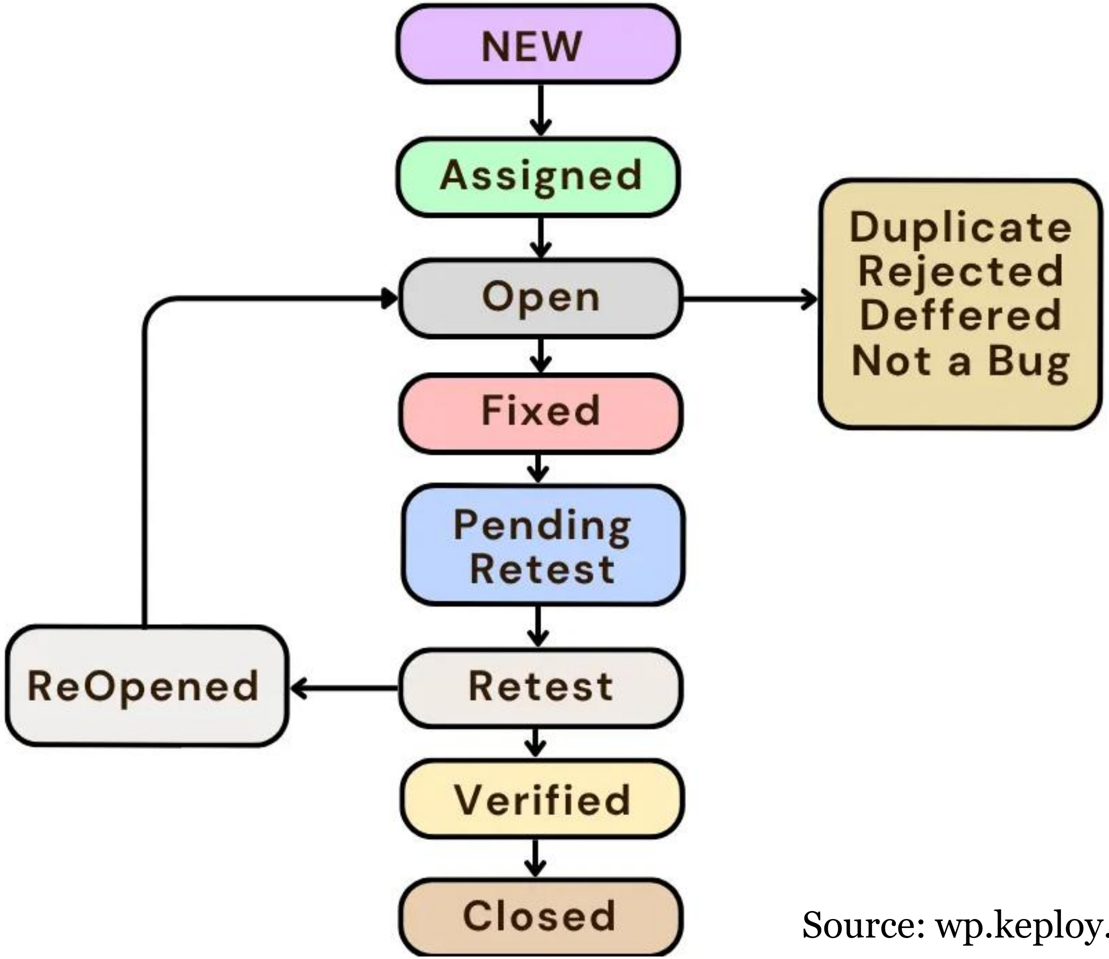


Test Case ID
Purpose
Preconditions
Inputs
Expected Outputs

Life cycle of a bug

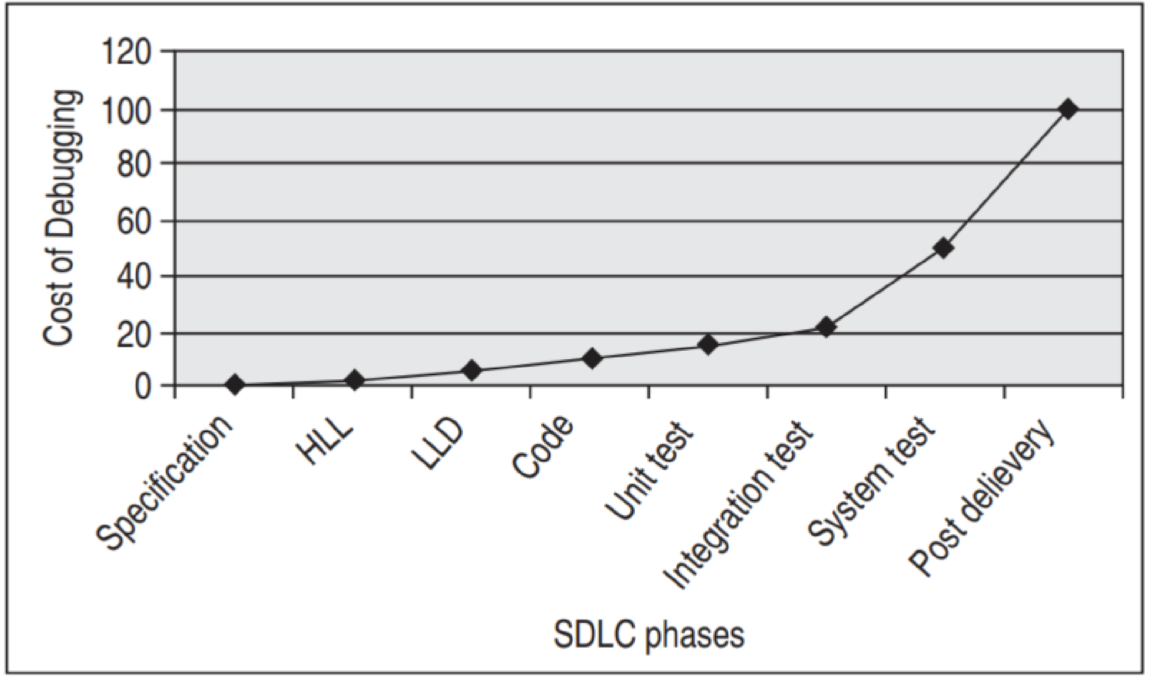


STATES OF A BUG

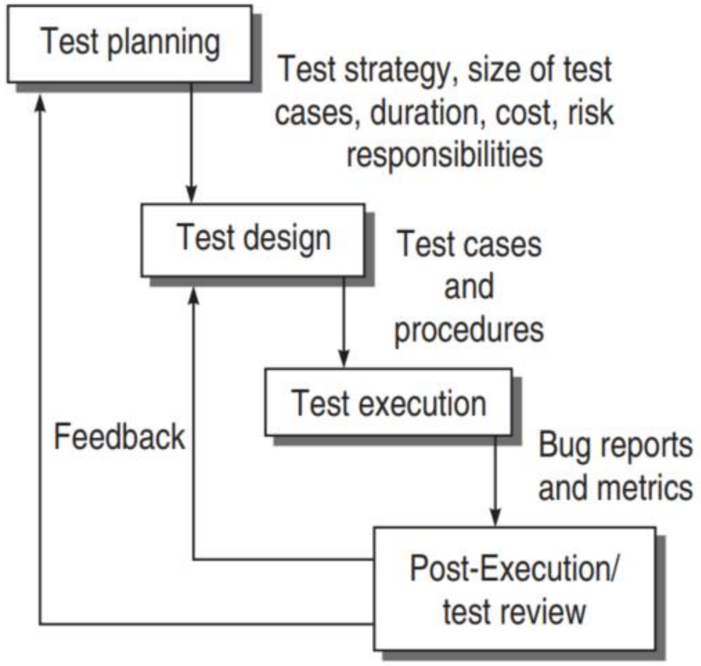


Source: wp.keploy.io/

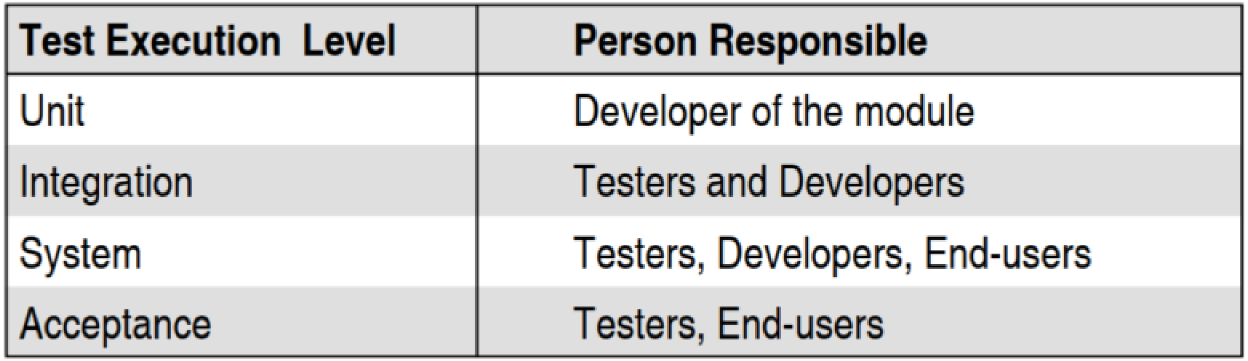
Cost of debugging if bug propagates



Software testing life cycle

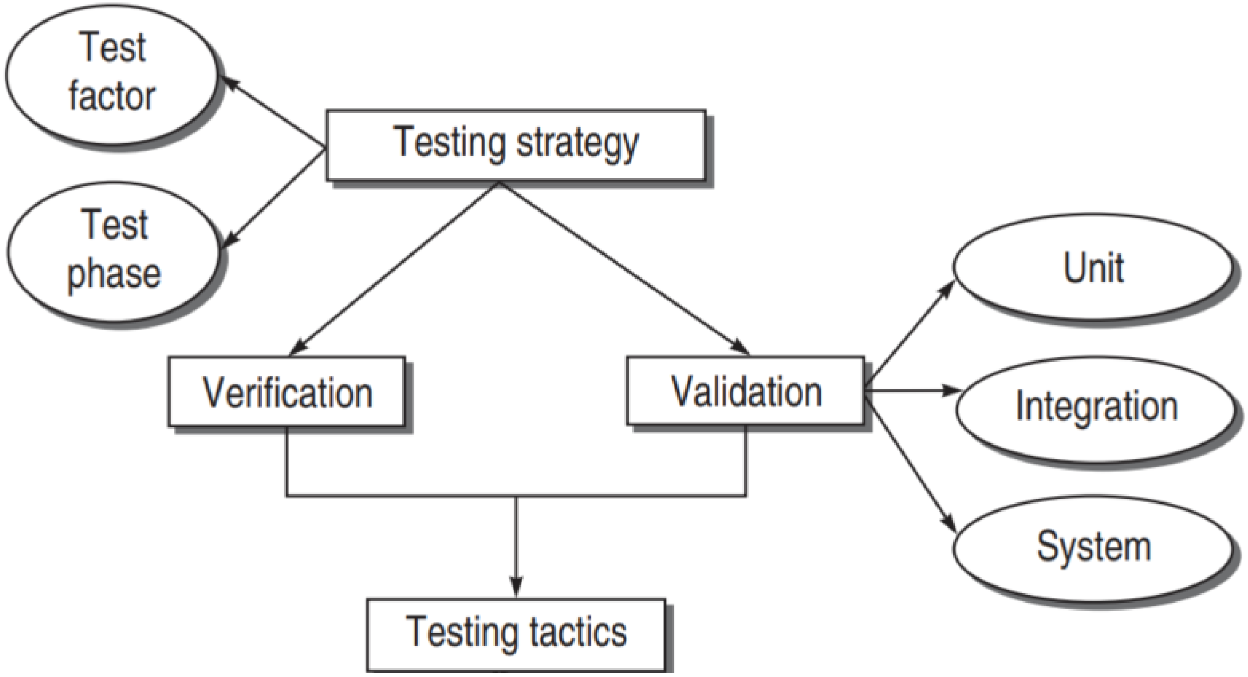

Test planning
Test strategy, size of test cases, duration, cost, risk responsibilities
Test design
Test cases and procedures
Test execution
Bug reports and metrics
Post-Execution/ test review
Feedback

Testing level vs responsibility

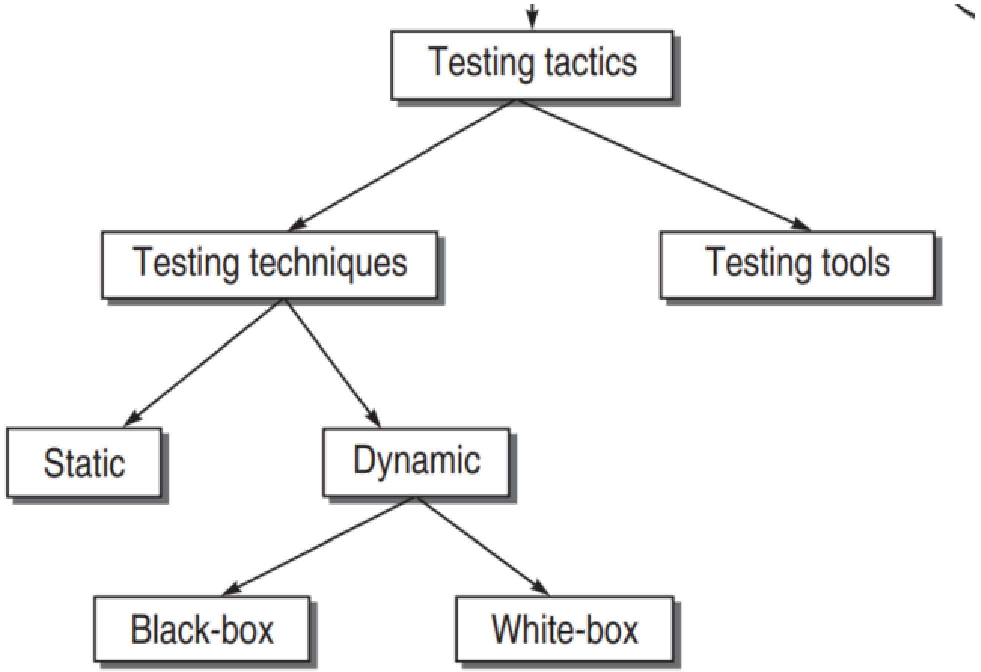


Test Execution Level	Person Responsible
Unit	Developer of the module
Integration	Testers and Developers
System	Testers, Developers, End-users
Acceptance	Testers, End-users

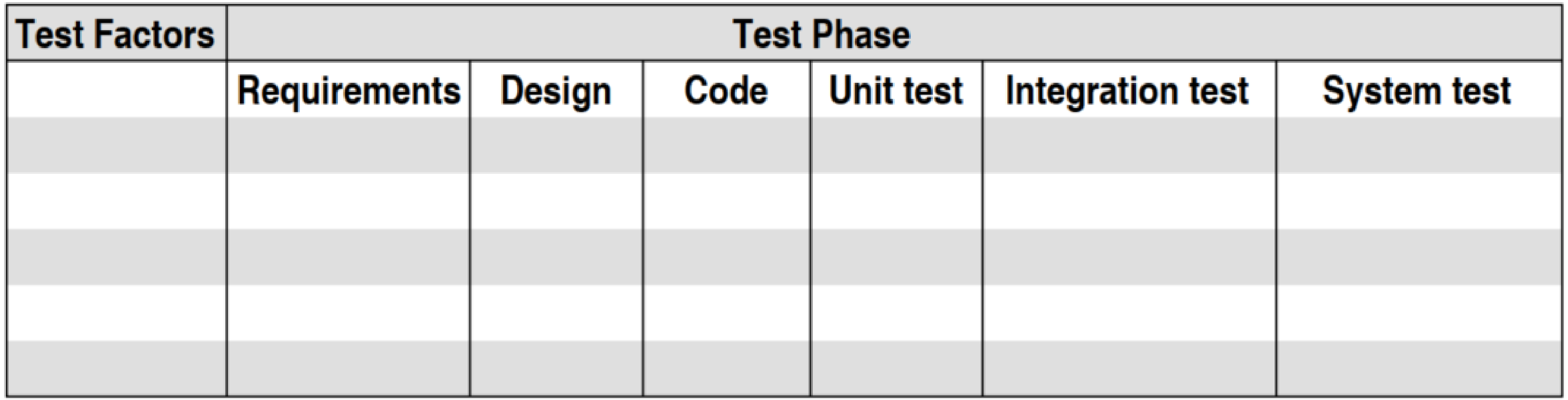
Testing methodology



Testing methodology

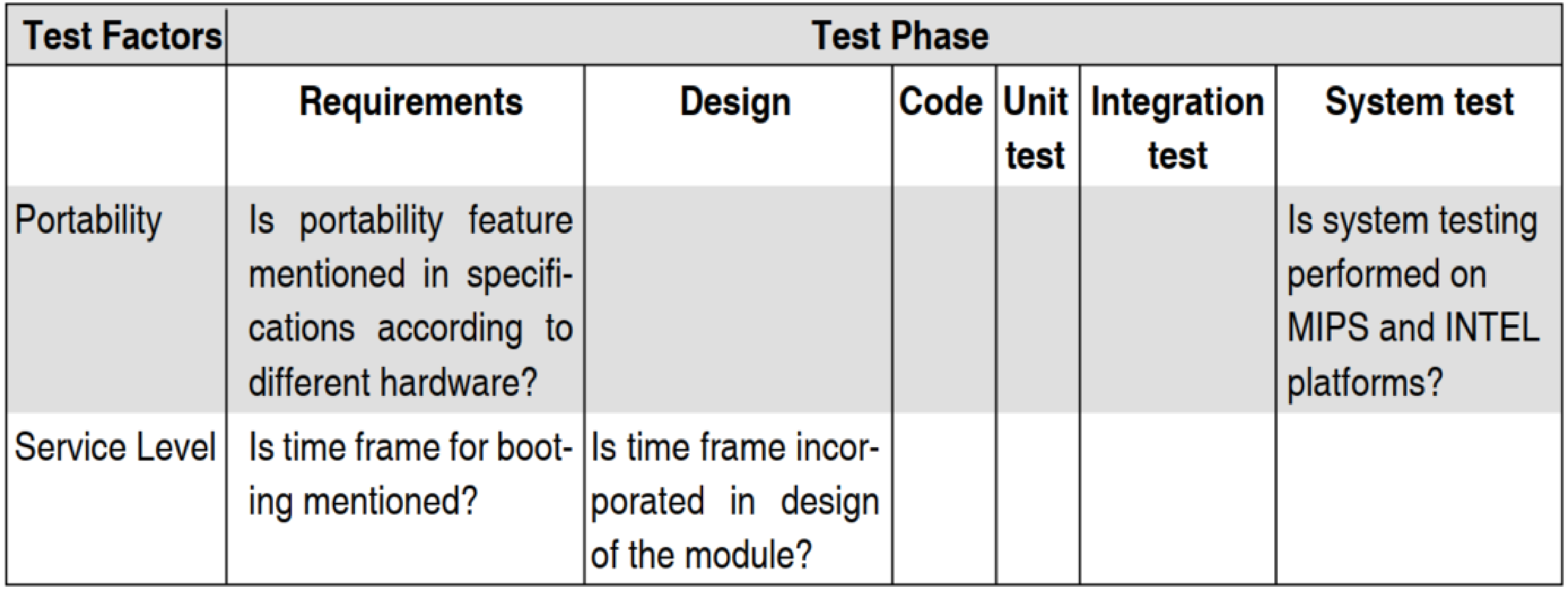


Test strategy matrix



Test Factors	Test Phase					
	Requirements	Design	Code	Unit test	Integration test	System test

Creating a test strategy



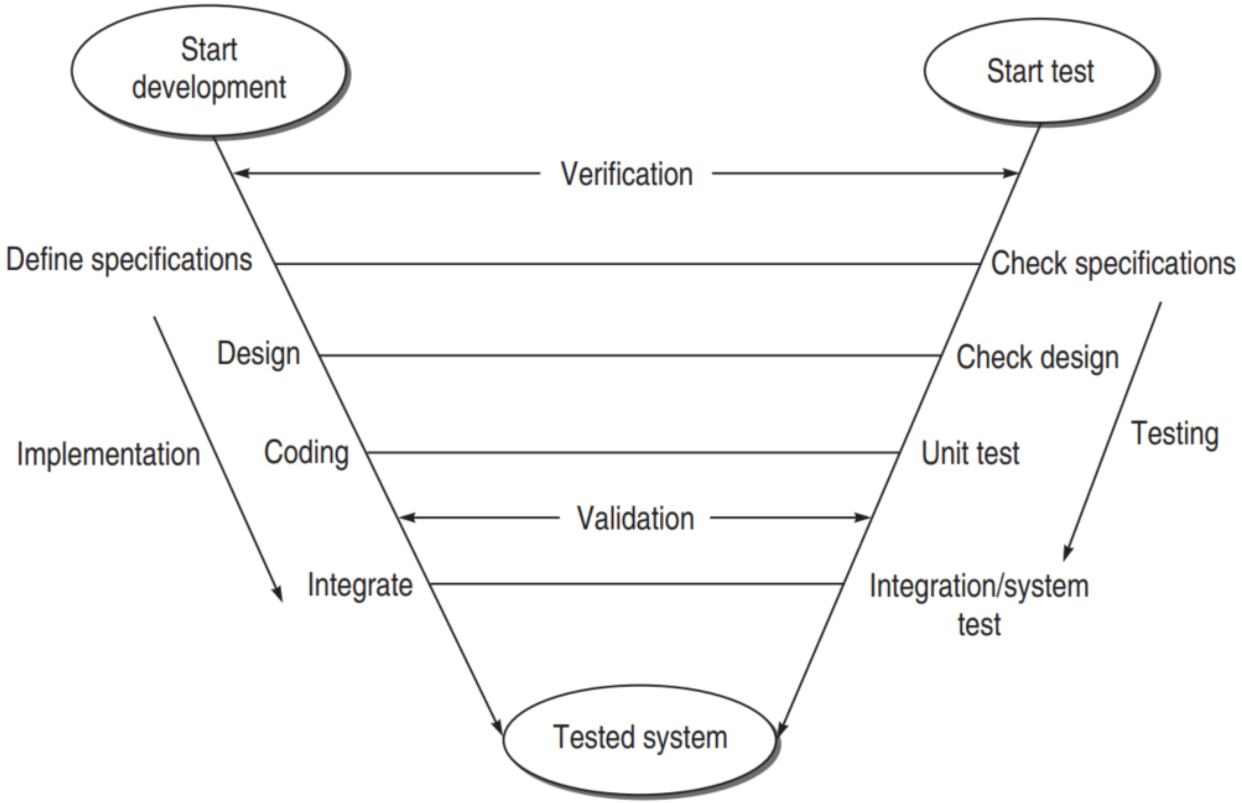
Test Factors	Test Phase					
	Requirements	**Design**	**Code**	**Unit test**	**Integration test**	**System test**
Portability	Is portability feature mentioned in specifi-cations according to different hardware?					Is system testing performed on MIPS and INTEL platforms?
Service Level	Is time frame for boot-ing mentioned?	Is time frame incor-porated in design of the module?				

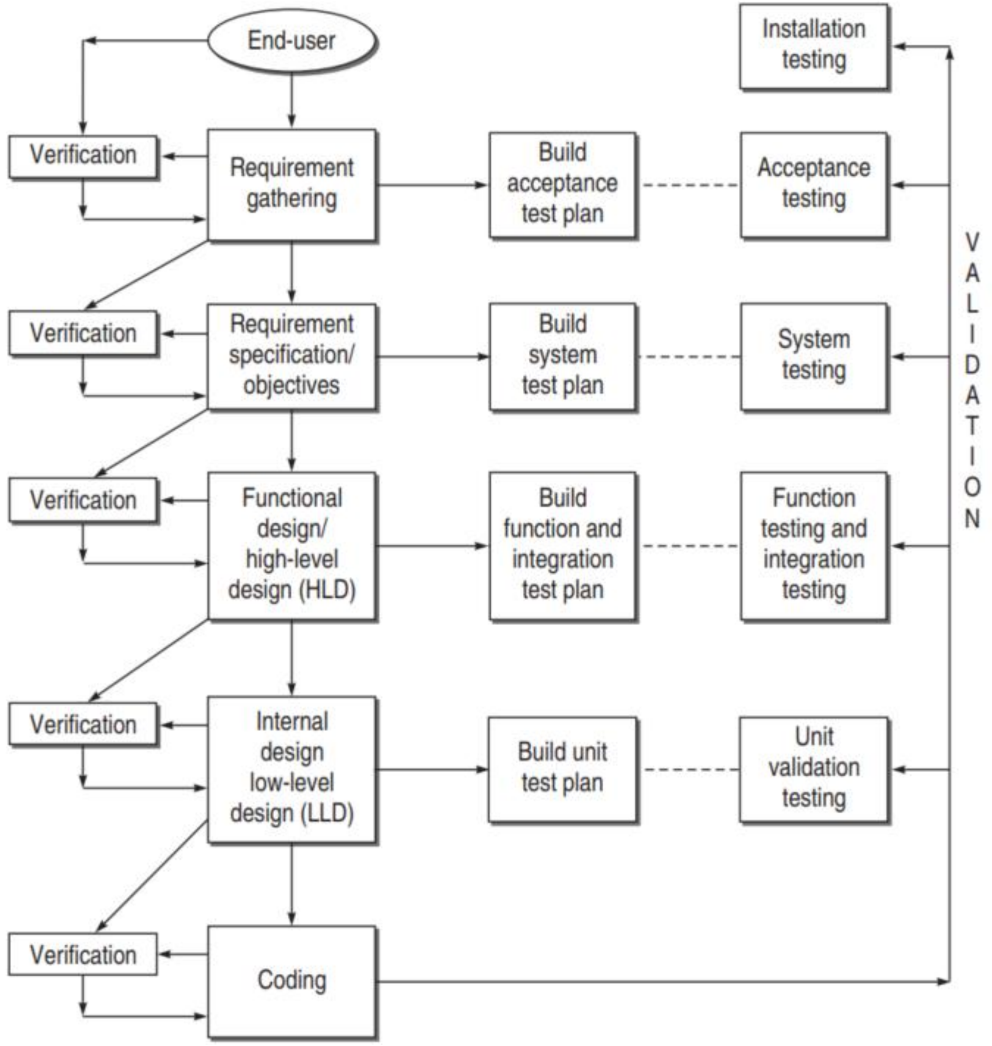
Verification and validation

- **Verification** refers to the process of evaluating a system or its components to ensure that the software correctly implements the specified requirements.
- It focuses on checking *whether the product is being built right*

- **Validation**, on the other hand, ensures that the developed software actually meets the user's needs and expectations.
- It focuses on checking *whether the right product is being built*.

Verification and validation Model




End-user
Verification
Requirement gathering
Build acceptance test plan
Acceptance testing
Installation testing
Verification
Requirement specification/ objectives
Build system test plan
System testing
Verification
Functional design/ high-level design (HLD)
Build function and integration test plan
Function testing and integration testing
Verification
Internal design low-level design (LLD)
Build unit test plan
Unit validation testing
Verification
Coding
VALIDATION

Verification and validation Diagram

How to verify and validate?

- Software is **verified** through activities such as reviews, inspections, and static analysis to ensure it meets specified design and requirement standards.
- Softeware is **validated** by performing functional and user acceptance testing to confirm that the software fulfills user needs and performs as intended by dynamic testing

Thank You