Software Testing

Ashima Tyagi
Assistant Professor
School of Computer Science & Engineering

Outline

- Software Testing
- Objectives of Software Testing
- Verification and Validation
- Levels of Testing

Software Testing

Software testing is the process of finding errors in the developed product.

Software testing can be stated as the process of verifying and validating whether a

- software or application is bug-free,
- meets the technical requirements as guided by its design and development, and
- meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases.

Objectives of Software Testing

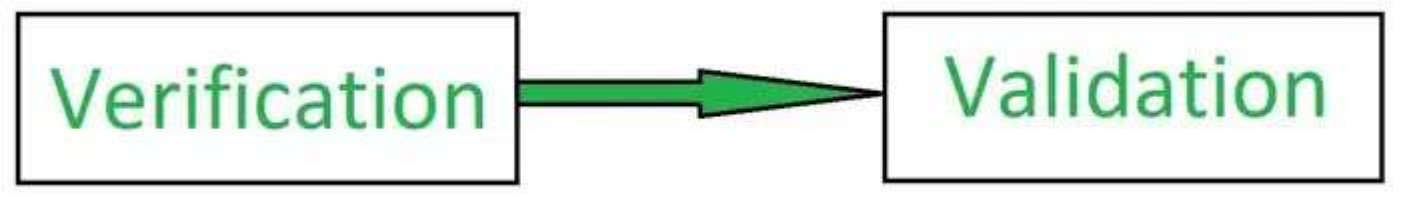
1. Identifying Defects in Software
2. Verifying Software Functionality
3. Validating User Requirements
4. Ensuring Compatibility and Integration
5. Assessing Performance and Security

Verification and Validation

Verification refers to the set of tasks that ensure that software correctly implements a specific function. Validation refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.

Verification: "Are we building the product right?"

Validation: "Are we building the right product?"



Verification Techniques:

1. Static Testing
2. Walkthroughs
3. Inspections
4. Reviews
5. Peer Reviews
6. Simulation
7. Prototyping
8. Dry Runs

Static Testing

- Static testing is a software testing method that examines a program -- along with any associated documents -- but does not require the program to be executed.
- Instead of executing the code, static testing is a process of checking the code and designing documents and requirements before it's run to find errors.
- The main goal is to find flaws in the early stages of development because it is normally easier to find the sources of possible failures this way.

Formal Technical Reviews

- **Formal Technical Reviews (FTRs) or Peer Reviews** are systematic, structured, and formal processes used in software engineering to evaluate the quality and correctness of software artifacts.
- These artifacts can include requirements, design documents, code, test plans, and other deliverables.
- The primary goal of FTRs is to identify defects, ensure compliance with standards, and improve the quality of the software product early in the development process.

Objectives of FTRs:

- Detect and Correct Defects Early
- Ensure Conformance to Standards
- Improve Software Quality
- Facilitate Knowledge Sharing
- Verify and Validate Requirements

Types of Formal Technical Reviews:

- Inspections:

A rigorous and detailed examination of software artifacts by a team of reviewers, usually led by a trained moderator. The process involves preparation, review meetings, and follow-up.

- Walkthroughs:

A less formal review where the author of the artifact presents it to a group of peers. The focus is on gaining feedback and suggestions rather than on defect detection alone.

- Audits:

An independent review conducted to ensure compliance with standards and procedures. Audits are often performed by external parties or internal quality assurance teams.

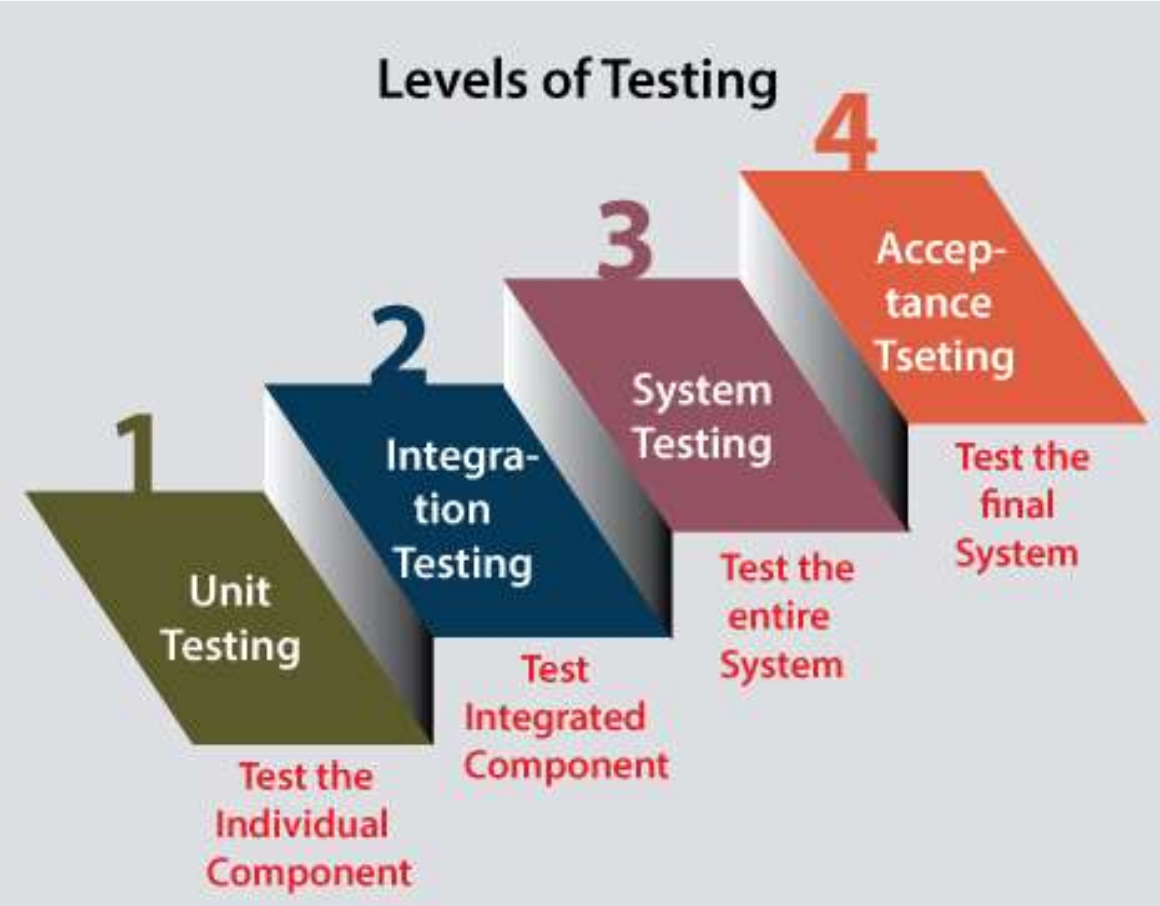
S.No.	Code Inspection	Walkthrough
1.	It is formal.	It is informal.
2.	Initiated by project team.	Initiated by author.
3.	A group of relevant persons from different departments participate in the inspection.	Usually team members of the same project take participation in the walkthrough. Author himself acts walkthrough leader.
4.	Checklist is used to find faults.	No checklist is used in the walkthrough.
6.	Formalized procedure in each step.	No formalized procedure in the steps.
7.	Inspection takes longer time as list of items in checklist is tracked to completion.	Shorter time is spent on walkthrough as there is no formal checklist used to evaluate program.
8.	Planned meeting with the fixed roles assigned to all the members involved.	Unplanned
9.	Reader reads product code. Everyone inspects it and comes up with detects.	Author reads product code and his teammate comes up with the defects or suggestions.
10.	Recorder records the defects.	Author make a note of defects and suggestions offered by teammate.
11.	Moderator has a role that moderator making sure that the discussions proceed on the productive lines.	Informal, so there is no moderator.

Validation Techniques:

1. Testing (Unit, Integration, System, Acceptance)
2. Black-Box Testing
3. White-Box Testing
4. Regression Testing
5. Alpha Testing
6. Beta Testing
7. User Acceptance Testing (UAT)
8. Stress Testing
9. Load Testing

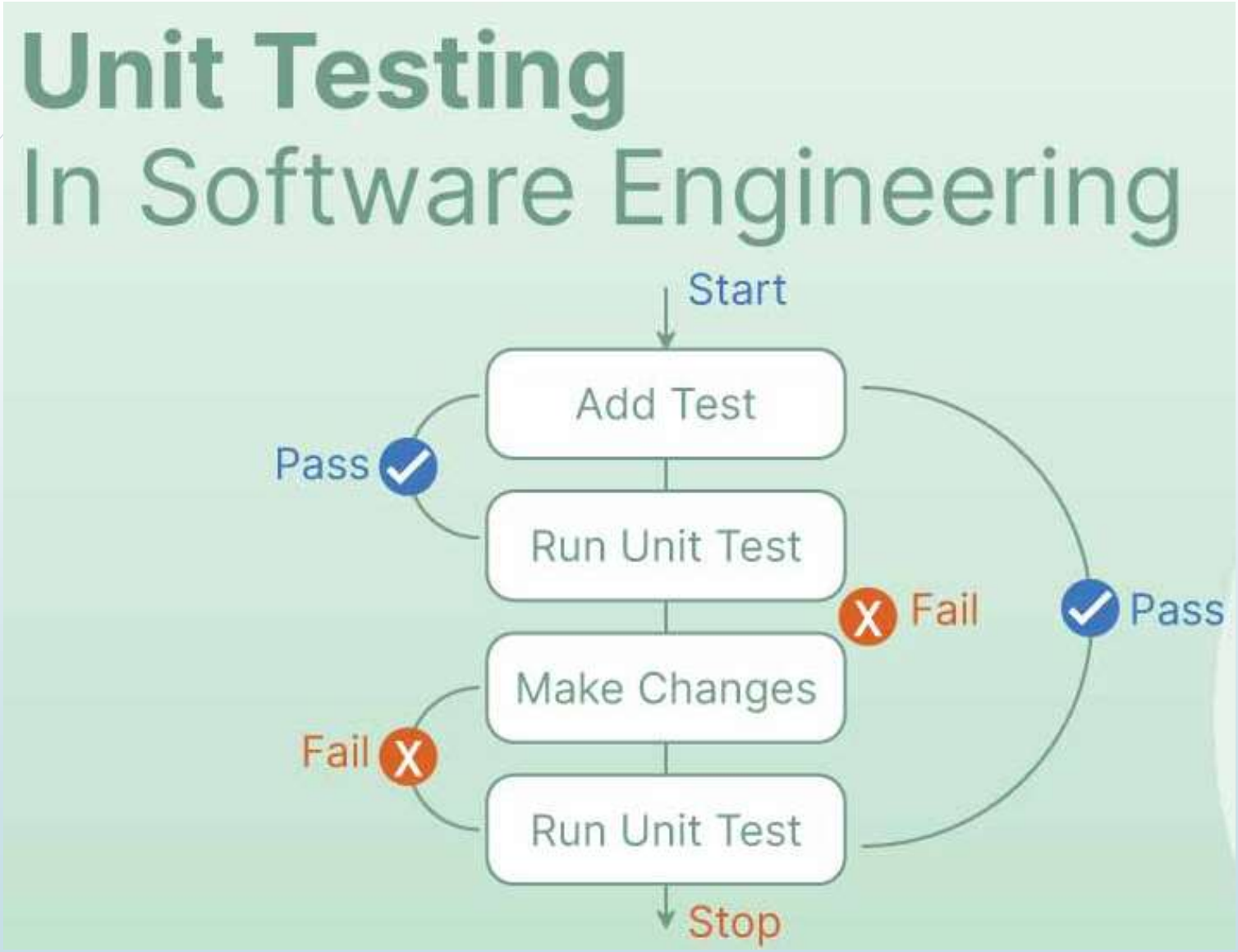
Levels of Testing

1. Unit Testing
2. Integration Testing
3. System testing
4. Acceptance Testing



1. Unit Testing

- Unit testing is a type of software testing that focuses on individual units or components of a software system.
- The purpose of unit testing is to validate that each unit of the software works as intended and meets the requirements.
- Developers typically perform unit testing, and it is performed early in the development process before the code is integrated and tested as a whole system.

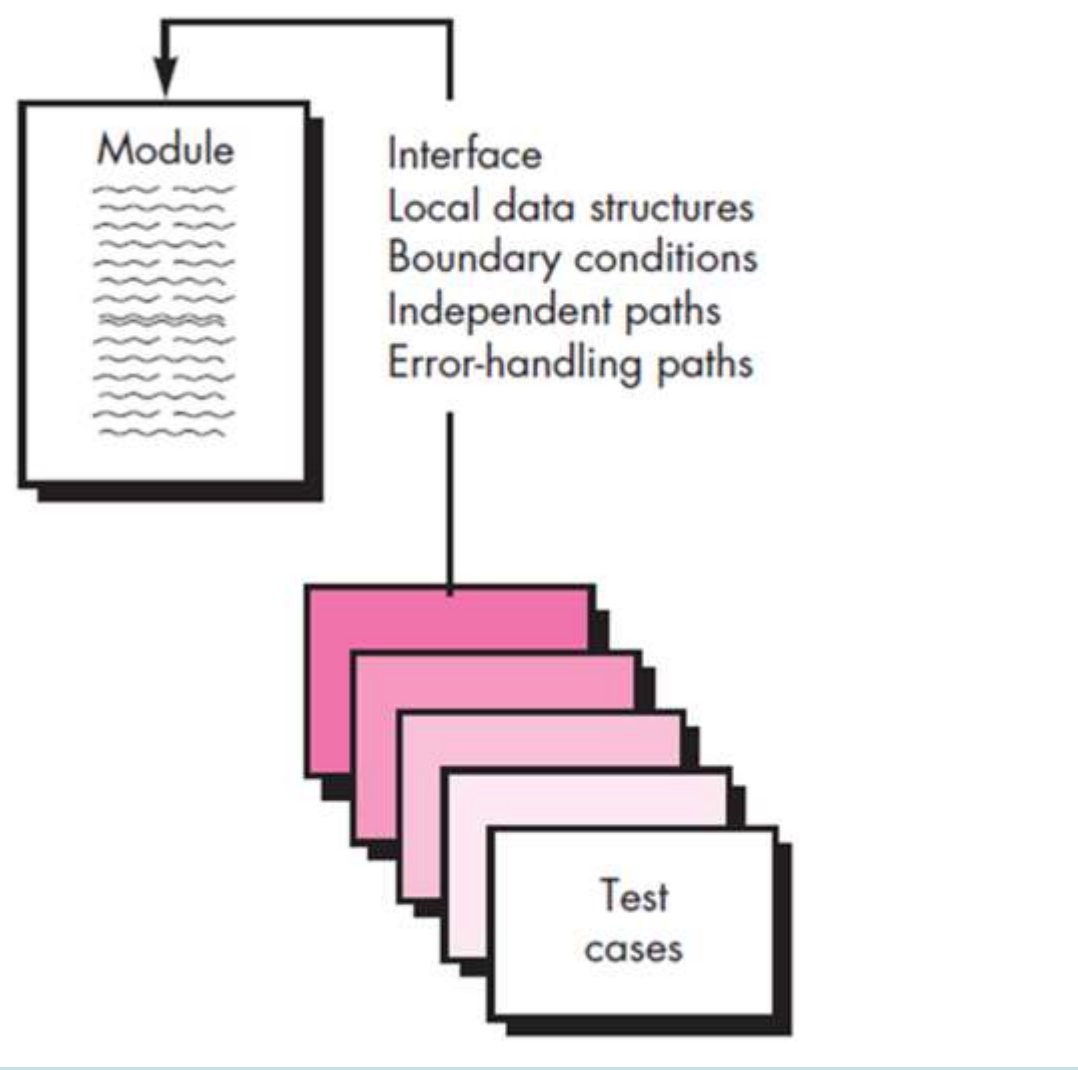

Unit Testing
In Software Engineering
Start
Add Test
Pass
Run Unit Test
Fail
Pass
Make Changes
Fail
Run Unit Test
Stop

Real life example

- Suppose a car manufacturer is making a car. He will make different parts like its engine, ignition system, wheels, etc. These different parts are the units that he will test individually. The manufacturer will check if the ignition system and the engine are working fine or not.
- Suppose the developer is making an e-commerce website. This website has modules like a login page, registration page, cart, payment page, dashboard, etc. The developer will check whether these modules are working fine or not. He will check whether the user can log in from Gmail, yahoo mail, etc. And in the same way for the payment page, he will check whether the user will be able to make payment through Gray, debit card, credit card, or cash on delivery, etc.

How unit testing works?

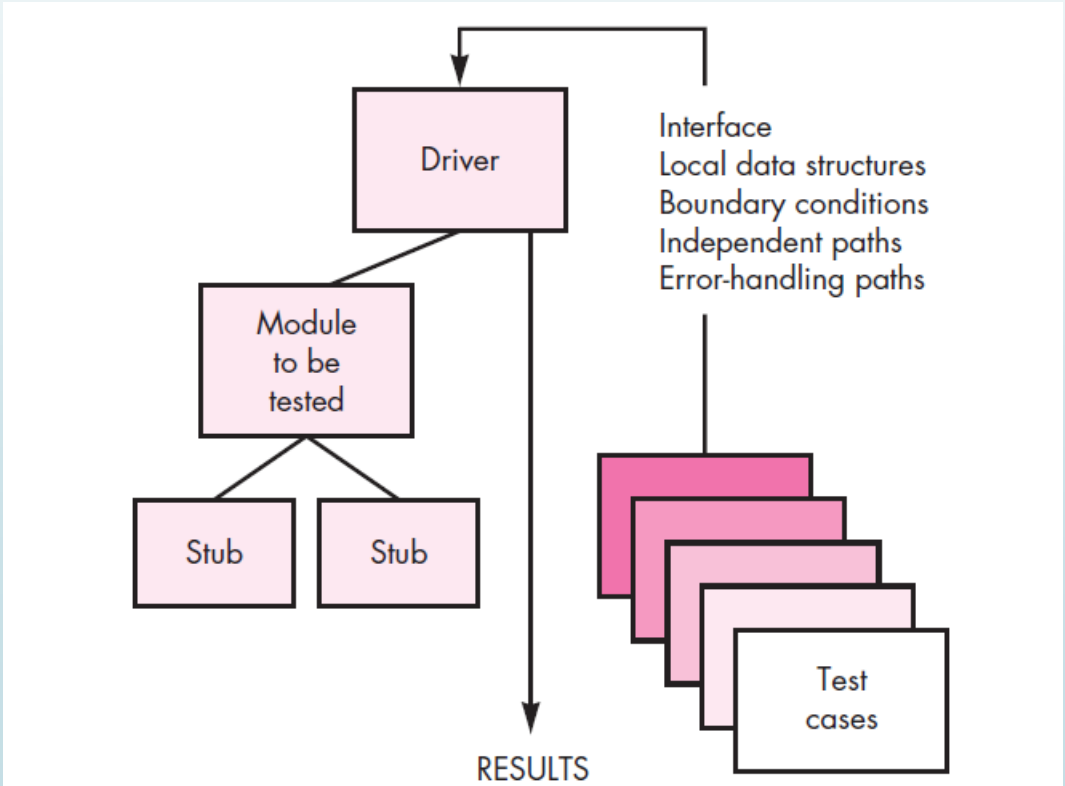
- You can first write unit tests as code.
- Then, run that test code automatically every time you make changes in the software code.
- This way, if a test fails, you can quickly isolate the area of the code that has the bug or error.



Strategies of Unit Testing:

- ***Interface Testing:*** Interface testing focuses on testing the interfaces of units, such as classes or modules, to ensure they conform to their specifications. This includes testing method signatures, parameter types, return types, and exception handling.
- ***Local Data Structures:*** Unit tests should test the behavior of local data structures used within units, such as arrays, lists, or maps. This includes testing edge cases and boundary conditions to ensure the data structure behaves as expected in all scenarios.
- ***Boundary Value Conditions***: Boundary value testing involves testing input values at or near the boundaries of valid ranges. This helps identify potential off-by-one errors or other issues that might occur at boundary conditions.
- ***Independent Paths***: Independent path testing aims to test all logical paths through a unit, ensuring that each path is exercised at least once. This helps uncover errors in conditional logic or branching statements.
- ***Error Handling:*** Error handling testing focuses on testing how a unit handles errors or exceptional conditions. This includes testing that exceptions are thrown when expected, and that error messages or recovery mechanisms are correct.

- Because a component is not a stand-alone program, driver and/or stub software must often be developed for each unit test.
- The unit test environment is illustrated in Figure.
- In most applications a driver is nothing more than a "main program" that accepts test case data, passes such data to the component (to be tested), and prints relevant results.
- Stubs serve to replace modules that are subordinate (invoked by) the component to be tested.

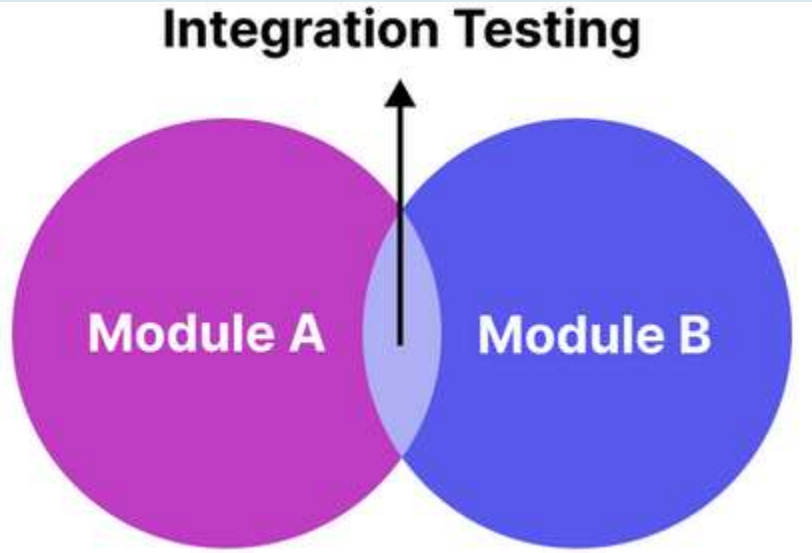


Purpose of Unit testing

1. Check code correctness.
2. Test all functions and operations.
3. Fix bugs early in the development cycle and save money.
4. To help developers understand the code base and make changes quickly.
5. Useful for code reuse.

2. Integration Testing

- Integration testing is the process of testing the interface between two software units or modules.
- The goal of integration testing is to identify any problems or bugs that arise when different components are combined and interact with each other.
- Once all the modules have been unit-tested, integration testing is performed.



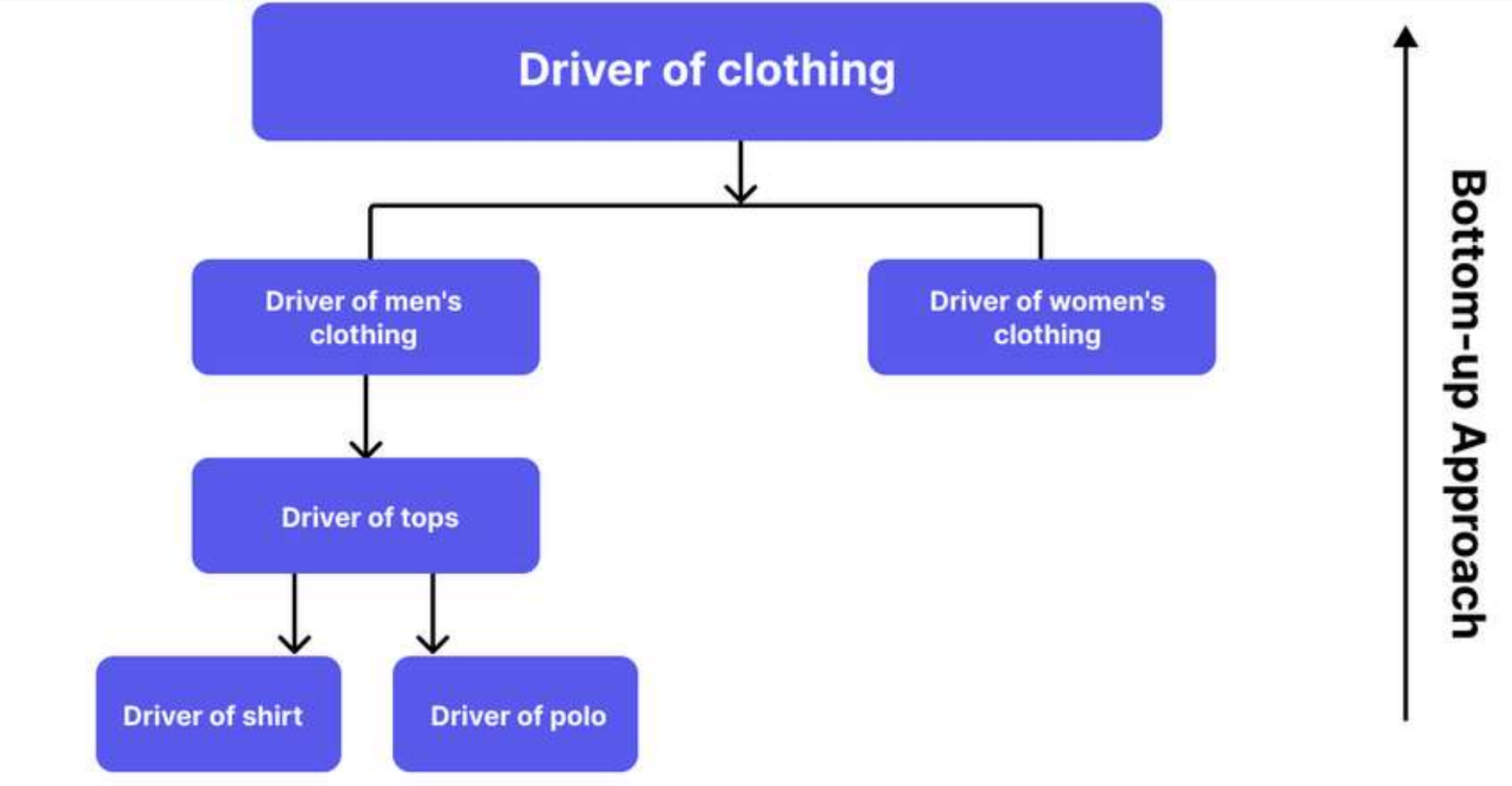
Integration testing is carried out by 2 different methods:

1. Bottom-up approach
2. Top-down approach

1. *Bottom-up approach*

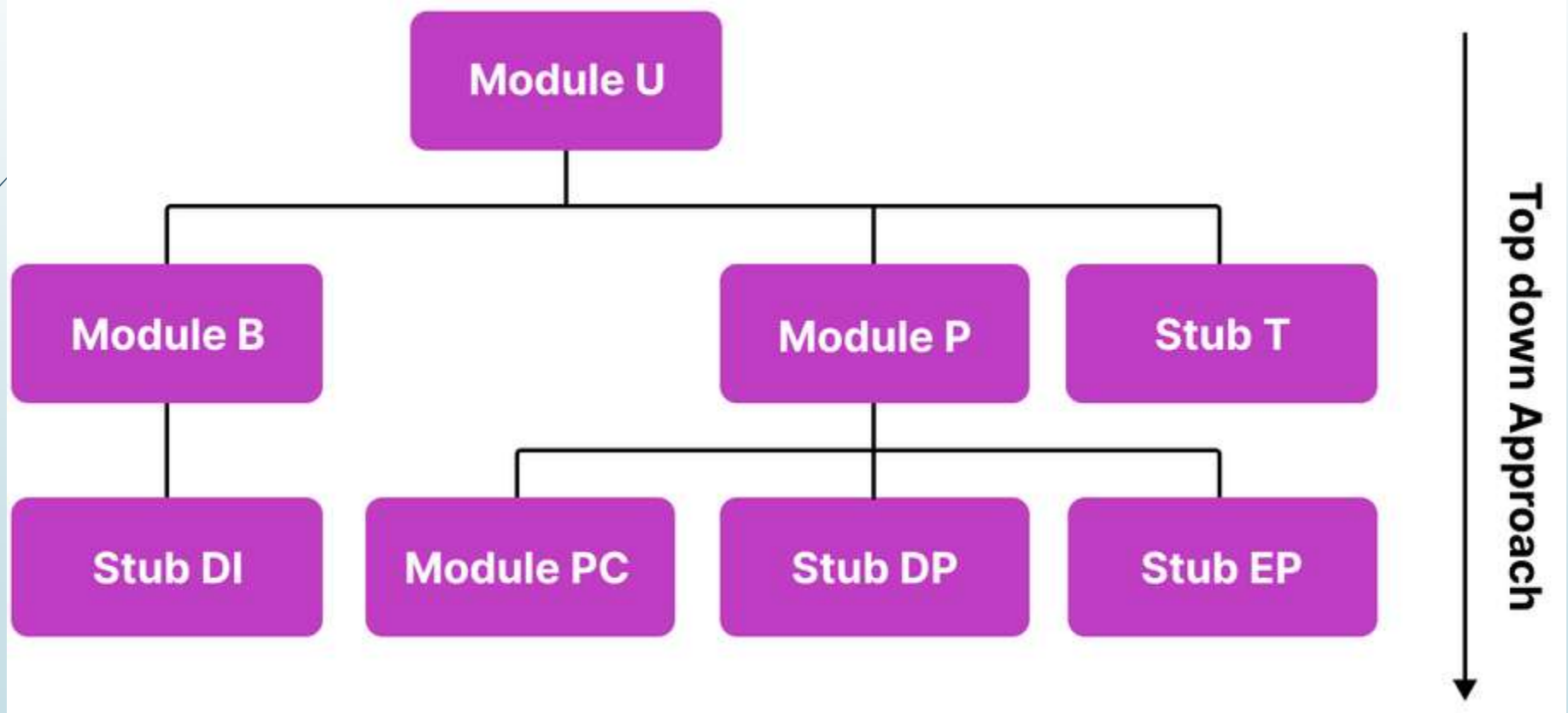
- With the bottom-up approach, testers start with individual modules at the lowest level, then gradually move to higher-level modules, hence the term "bottom-up".
- The rationale for this approach is that the entire system can only work reliably if these basic building blocks work reliably.

- A "bottom-up" approach is essentially going from the more specific and granular components to more general and comprehensive components.



2. *Top-Down approach*

- With the top-down approach, testers start with the highest-level modules, then gradually move to lower-level modules, hence the term "top-down".



We should use bottom-up integration testing when:

- Complexity is primarily found in lower-level modules
- The team follows incremental development (which involves developing lower-level components first before moving to higher-level modules)
- Defect localization is a crucial aspect of the project since the granularity of the bottom-up approach provides more exact bug isolation
- Higher-level modules are still under development or likely to change frequently

Similarly, we should use top-down integration testing when:

- The critical functionalities are primarily concentrated in the higher-level components
- When it's crucial to simulate real-world scenarios and user interactions, top-down testing provides a more holistic view of the system's behavior early on.
- Lower-level modules are more well-defined, stable, and unlikely to change significantly
- Top-down testing allows for faster validation of user-facing features, enabling early prototypes for user feedback.

Note that when choosing the bottom-up testing approach, we may need to build a stub to substitute for high-level modules that are not yet developed, while for the top-down testing approach, we may need to build a driver to replace the unavailable low-level modules.

3. System Testing

- System testing is a type of software testing that evaluates the overall functionality and performance of a complete and fully integrated software solution.

Types of System Testing

1. **Performance Testing:** Performance Testing is a type of software testing that is carried out to test the speed, stability and reliability of the software product or application.
2. **Load Testing:** Load Testing is a type of software Testing which is carried out to determine the behavior of a system or software product under extreme load. Ex: Simulating **1,000 concurrent users** on a website to check if it performs as expected.
3. **Stress Testing:** Stress Testing is a type of software testing performed to check the robustness of the system under the varying loads. stress tests apply extreme load to determine the point at which your system breaks due to high traffic and heavy usage. Ex: Increasing traffic beyond the system's maximum capacity to see when it crashes.
4. **Scalability Testing:** Scalability Testing is a type of software testing which is carried out to check the performance of a software application or system in terms of its capability to scale up or scale down the number of user request load.

4. Acceptance Testing

- Acceptance testing is formal testing based on user requirements and function processing.
- It determines whether the software is conforming specified requirements and user requirements or not.

Once the application is bug-free, we handover it to the customer, no customer accept the application blindly before using it. Hence, they do one round of testing for their satisfaction, which is known as User Acceptance Testing (UAT).

Types of User Acceptance Testing (UAT).

- Alpha testing
- Beta testing

Alpha Testing

- Alpha testing is a type of acceptance testing, which is performed to identify all possible bugs/issues before releasing the product to the end-user.
- Alpha test is a preliminary software field test carried out by a team of users to find out the bugs that were not found previously by other tests.
- Alpha testing needs lab environment, and usually, the testers are an internal employee of the organization.
- This testing is called alpha because it is done early on, near the end of the software development, but before beta testing.

Beta Testing

- Beta testing of a product is implemented by "real users "of the software application in a "real environment."
- In this phase of testing, the software is released to a limited number of end-users of the product to obtain feedback on the product quality.
- It allows the real customers an opportunity to provide inputs into the design, functionality, and usability of the product.

Thank You