EAST WEST UNIVERSITY

CSE 412

Software Engineering

Yasin Sazid
Lecturer
Department of CSE
East West University

Topic 6: Software Testing

Software Testing

- Testing is defined as the execution of a program to find its faults
- While more time is typically spent on testing than in any other phase of SDLC, there is considerable confusion about its purpose
- Many software professionals, for example, believe that tests are run to show that the program works rather than to learn about its faults

How to define Software Testing Principles

- Testing:

  The execution of a program to find its unexpected behavior
- Verification:

  The process of proving the programs correctness.
- Validation:

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

  Diagnosing the error and correct it

Validation vs Verification

Aspect	**Verification**	**Validation**
Purpose	Ensure the product is being built correctly (according to specifications)	Ensure the right product is built (according to user needs)
Question	Are we building the product right?	Are we building the right product?
Focus	Process-oriented: correctness of intermediate work-products	Product-oriented: usefulness and satisfaction of the final system
When	During development (before execution/testing)	After development (during/after testing, in real use conditions)

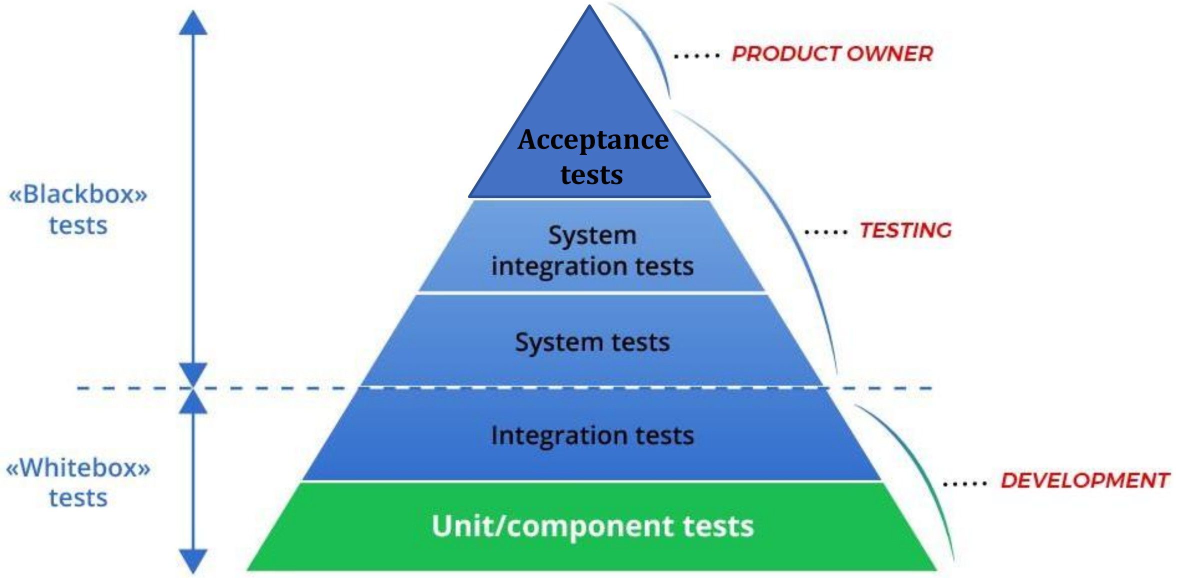
Software Testing Principles

- To remove as many defects as possible before test since the quality improvement potential of testing is limited
- Exhaustive testing is practically impossible
  - Example: 10-letter string $\rightarrow 26^{10}$ combinations
  - Testing 1 input per microsecond = ~4.5 million years needed!
- Reality
  - We can only test a small, smart subset of conditions that will reveal the characteristics of the program
- Therefore
  - Focus on early defect removal (reviews, verification)
  - Testing is for detection, not perfection

Types of Software Tests

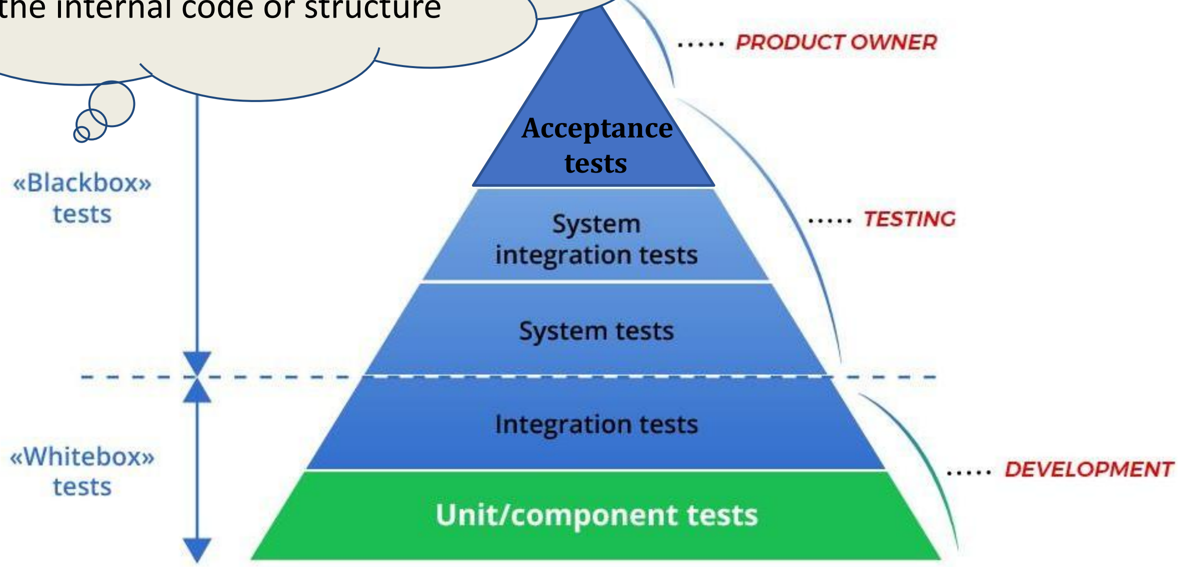
- Unit Testing
- Integration Testing
- Function Testing
- Regression Testing
- System Testing
- Acceptance and Installation Tests

Types of Software Testing

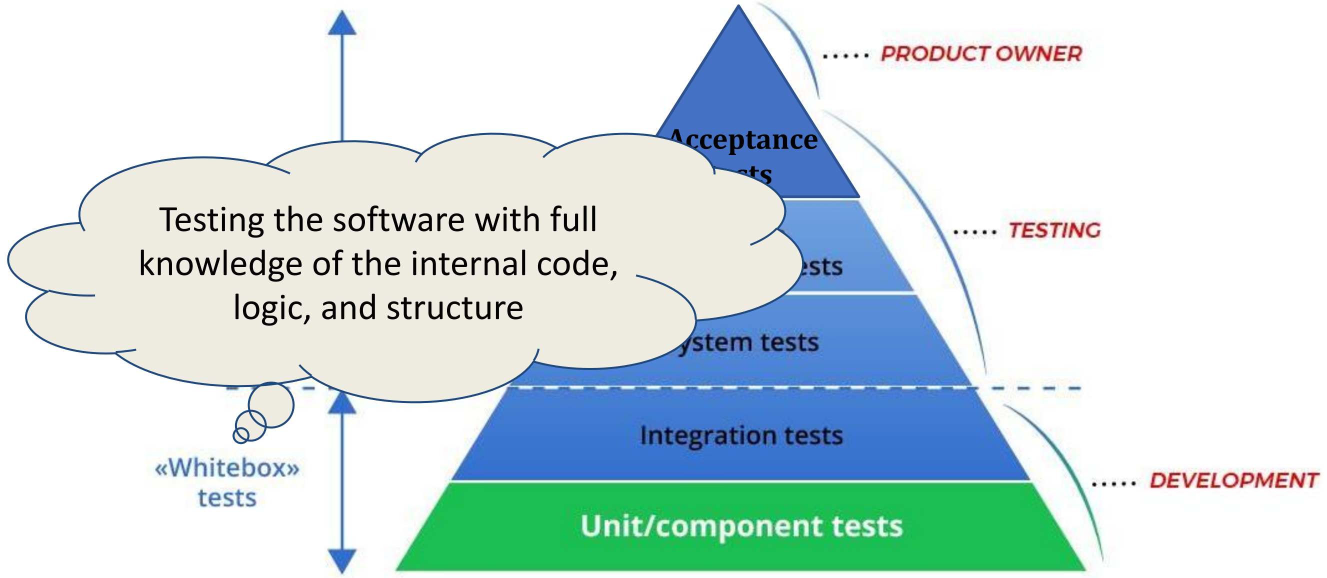


Types of Software Testing

Testing the software's functionality without knowing the internal code or structure



Types of Software Testing



Testing the software with full knowledge of the internal code, logic, and structure

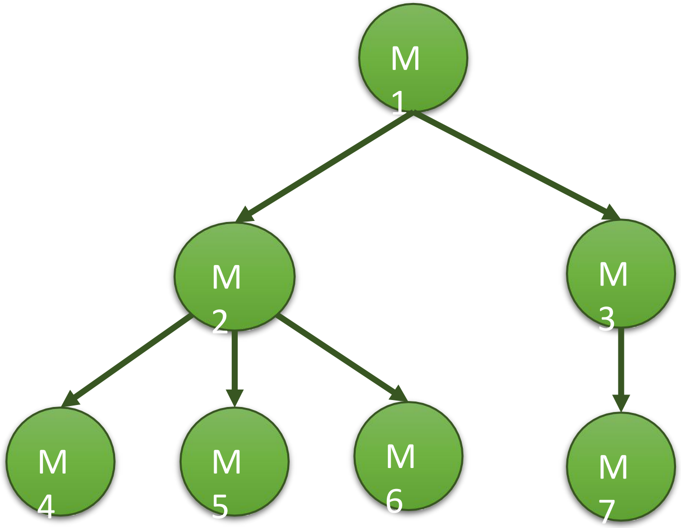
Unit Testing

- 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
- Disadvantage: The tester may be biased by previous experience. And the test value may not cover all possible values
- While its disadvantages are significant, white box testing generally has the highest error yield of all testing techniques

Integration Testing

- This phase involves testing of modules which have been integrated in sub-system. A module is a collection of dependent components such as object class, and abstract data type of some looser collection of procedures and functions.
- On very large system it is often wise to do integration testing in several steps. Such systems generally have several relatively large components that can be built and integrated separately before combination into a full system.
- Integration Testing is divided into -
  - Top-down Integration Test
  - Bottom-up Integration Test

Top-down Testing

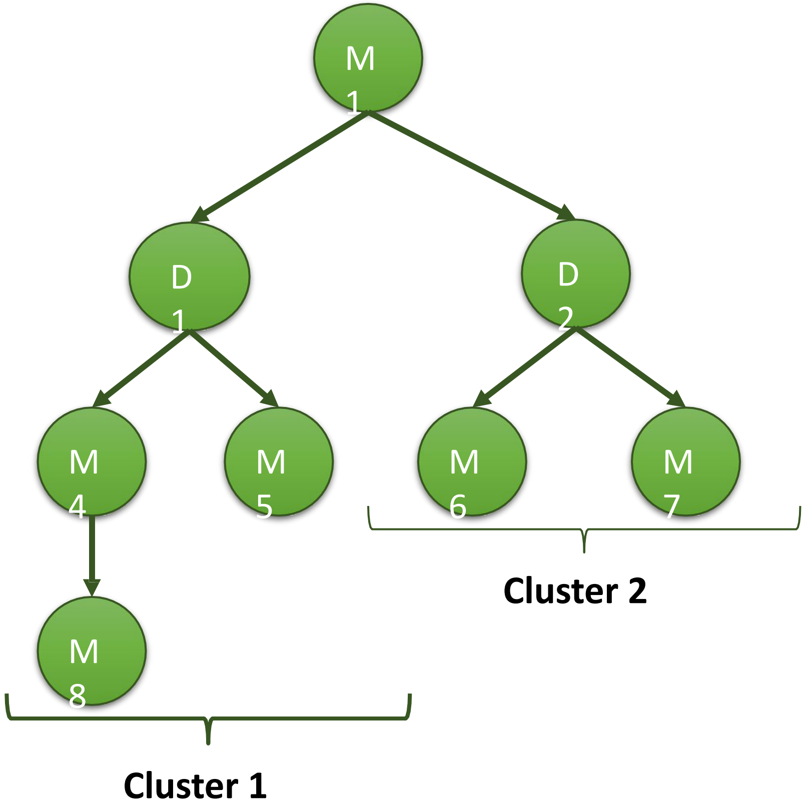


- In the top-down integration testing, if depth-first approach is adopted then we will start integration from module M1. Then we will integrate M2, then M3, M4, M5, M6, and at last M7.
- In the top-down integration testing, if breadth-first approach is adopted, then we will integrate module M1 first, then M2, M6. Then we will integrate module M3, M4, M5, and at last M7

Top-down Integration Test

- The control program is tested first
- Modules are integrated one at a time
- Emphasize on interface testing
- Advantages:
  - There is no need to write drivers
  - Interface errors are identified at an early stage and fault localization is also easier
  - Low-level utilities that are not important are not tested well and high-level testers are tested well in an appropriate manner
  - Representation of test cases is easier and simple once Input-Output functions are added
- Disadvantages:
  - It requires a lot of stubs and mock objects
  - Representation of test cases in stubs can be not easy and might be difficult before Input-Output functions are added
  - Low-level utilities that are important are also not tested well

Bottom-up Testing



Bottom-up Integration Test

- Allow early testing aimed at proving feasibility
- Emphasize on module functionality and performance
- Advantages:
  - It is easy and simple to create and develop test conditions
  - It is also easy to observe test results
  - It is not necessary to know about the details of the structural design
  - Low-level utilities are also tested well and are also compatible with the object-oriented structure
- Disadvantages:
  - Towards top of the Hierarchy, it becomes very complicated
  - There is no concept regarding early skeletal system
  - There will be an impact on sibling and higher-level unit tests due to changes

Function Testing

- Designed to exercise the program according to its external specifications
- Testers not biased by knowledge of the program’s design
- Disadvantages:
    - The need for explicitly stated requirements
    - Only cover a small portion of the possible test conditions
- Since exhaustive black box testing is generally impossible, these tests should be viewed as statistical sampling -
    - when errors are found, a closer examination is required. Functional testing starts by examining the functions the program is to perform and devising a sequence of inputs to test them

Regression Testing

- Regression Testing is the process of testing the modified parts of the code and the parts that might get affected due to the modifications to ensure that no new errors have been introduced in the software after the modifications have been made.
- Regression means return of something and in software testing, it refers to the return of a bug.
- Firstly, whenever some changes are made to the source code for any reasons like adding new functionality, optimization, etc. then the program when executed fails in the previously designed test suite for obvious reasons.
- After the failure, the source code is debugged in order to identify the bugs in the program. After identification of the bugs in the source code, appropriate modifications are made.
- Then appropriate test cases are selected from the already existing test suite which covers all the modified and affected parts of the source code.
- We can add new test cases if required. In the end regression testing is performed using the selected test cases.

Regression Testing

- The progressive phase introduces and tests new functions, uncovering problems in the newly added or modified modules and in their interfaces with the previously integrated modules.
- The regressive phase concerns the effect of the newly introduced changes on all the previously integrated code. Problems arise when errors made in incorporating new functions affect previously tested functions.
- The basic regression testing approach is to incorporate selected test cases into a regression test bucket that is run periodically in an attempt to detect regression problems.
  - The full bucket is run only occasionally, but the subset is run against every spin.
  - The spin subset should include all the test cases for any recently integrated functions and a selected sample from the full regression bucket.

Regression Testing

- Advantages:
  - It ensures that no new bugs has been introduced after adding new functionalities to the system.
  - As most of the test cases used in Regression Testing are selected from the existing test suite and we already know their expected outputs. Hence, it can be easily automated by the automated tools.
  - It helps to maintain the quality of the source code.
- Disadvantages:
  - To decide how much of a subset to use and which tests to select.
  - It can be time and resource consuming if automated tools are not used.
  - It is required even after very small changes in the code.

Smoke testing

- Smoke testing is an integration testing approach that is commonly used when product software is developed. It is designed as a pacing mechanism for time-critical projects, allowing the software team to assess the project on a frequent basis.
- The smoke test should exercise the entire system from end to end. It does not have to be exhaustive, but it should be capable of exposing major problems. The smoke test should be thorough enough that if the build passes, you can assume that it is stable enough to be tested more thoroughly.
- A series of tests is designed to expose errors that will keep the build from properly performing its function. The intent should be to uncover "showstopper" errors that have the highest likelihood of throwing the software project behind schedule.

Types of Smoke Testing

- **Manual smoke testing:** Human software testers conduct smoke tests manually. This includes manually developing and updating test cases. Test scripts are also written manually for new or existing features.
- **Automated smoke testing:** Software tools are used to automate the smoke testing process. Smoke testing tools make the testing process more efficient by automatically providing relevant tests.
- **Hybrid smoke testing:** Hybrid testing is a combination of manual and automated smoke testing, where testers write test cases and then automate the tests using a tool.

System Testing

- **Recovery testing** is a system test that forces the software to fail in a variety of ways and verifies that recovery is properly performed. If recovery is automatic, reinitialization, checkpointing mechanisms, data recovery, and restart are evaluated for correctness. If recovery requires human intervention, the mean-time-to-repair (MTTR) is evaluated to determine whether it is within acceptable limits.
- **Security testing** attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration.
- **Stress testing** executes a system in a manner that demands resources in abnormal quantity, frequency, or volume

Validation Testing

- QA teams conduct validation testing to ensure the software meets the business and user requirements and performs as intended in real-world scenarios
- Key Activities:

  Validate functionality against user requirements

  Test user workflows and real-world use cases

  Ensure system usability and business process alignment

Acceptance Testing

- A process to verify that a software system meets business requirements and is ready for deployment, typically performed by end users or stakeholders to ensure the product fulfills user needs and expectations.
- **Alpha Testing:** The alpha test is conducted at the developer's site by a representative group of end users. The software is used in a natural setting with the developer "looking over the shoulder" of the users and recording errors and usage problems. Alpha tests are conducted in a controlled environment.
- **Beta Testing:** The beta test is conducted at one or more end-user sites. Unlike alpha testing, the developer generally is not present. Therefore, the beta test is a "live" application of the software in an environment that cannot be controlled by the developer

THANK YOU