Software Engineering
(CSM 31212)

Lecture 09:Software Testing

MM. Mohamed Mufassirin

Lecturer in Computer Science

Dept. of Computer Science

FAS/SEUSL

Validations and Verification

- Validation and verification (V & V) is the name given to the checking and analysis processes that ensure that software conforms to its specification and meets the needs of the customers who are paying for the software.
- V & V is a whole life-cycle process. It starts with requirements reviews and continues through design reviews and code inspections to product testing. There should be V& V activities at each stage of software process.
- Validation : Are we building the right product?

Verification : Are we building the product right?

Validation and Verification…..

Within the V & V process, two techniques of system checking and analysis may be used:

1. ***Software Inspections***
2. **Software Testing**

Software Inspections- Analyse and check system representations such as the requirements documents, design diagrams and program source code.

- They may be applied at all stages of the development process.
- Inspections may be supplemented by some automated analysis of the source text of a system or associated documents.
- Software inspections and automated analysis are static V & V techniques as they do not require the system to be executed.

Validation and Verification…..

Software Testing - involves executing an implementation of the software with test data and examining the outputs of the software and its operational behaviour to check that it is performing as required.

- Testing is a dynamic technique of V & V because it works with an executable representation of the system.

Software Testing Procedure

- Testing procedures should be established at the start of any software project. All testing carried out should be based on a test plan, this should detail which tests are to be carried out. For each test, the following information should be included in the test plan:

  * The pre-requisites for the tests.
  * The steps required to carry out the tests
  * The expected results of the test.

The outcome of any tests should be recorded in a test results document that include whether the test succeeded or failed and a description of the failure. Test results for all passes through the test plan must be recorded to allow accurate records to be kept of where problems occur and when they were identified and corrected.

What is a Test Case?

A test case is typically defined as a document that lays out the following:

- Test data
- Procedures/inputs
- Scenarios
- Descriptions
- Testing environment
- Expected results
- Actual results

- It's used to for a singular test scenario. As a rule, there are usually both a positive and negative test case for each scenario.
- The purpose of a test case is two-fold: It's designed to find any errors or bugs within the software application, and it's also designed to show how the application should be executed if it performs correctly.
- It also demonstrates real-world use of the product and whether it fits the customer needs.

Test Case Template

Project Name:

Test Case Template

Test Case ID: Fun_10	**Test Designed by:** <Name>
Test Priority (Low/Medium/High): Med	**Test Designed date:** <Date>
Module Name: Google login screen	**Test Executed by:** <Name>
Test Title: Verify login with valid username and password	**Test Execution date:** <Date>
Description: Test the Google login page	

Pre-conditions: User has valid username and password
Dependencies:

Step	Test Steps	Test Data	Expected Result	Actual Result	Status (Pass/Fail)	Notes
1	Navigate to login page	User= example@gmail.com	User should be able to login	User is navigated to	Pass	
2	Provide valid username	Password: 1234		dashboard with successful		
3	Provide valid password			login		
4	Click on Login button					

Post-conditions:
User is validated with database and successfully login to account. The account session details are logged in database.

Test Case: Example

Test Case Example2 (complex test may be divided into more than test cases)

Test Case #: 2.3	**Test Case Name:** Change PIN	**Page:** 1 of 1
System: ATM	**Subsystem:** PIN	
Designed by: ABC	**Design Date:** 28/11/2004	
Executed by:	**Execution Date:**	
Short Description: Test the ATM Change PIN service		

Pre-conditions
The user has a valid ATM card - The user has accessed the ATM by placing his ATM card in the machine
The current PIN is 1234
The system displays the main menu

Step	Action	Expected System Response	Pass/ Fail	Comment
1	Click the 'Change PIN' button	The system displays a message asking the user to enter the new PIN		
2	Enter '5555'	The system displays a message asking the user to confirm (re-enter) the new PIN		
3	Re-enter '5555'	The system displays a message of successful operation The system asks the user if he wants to perform other operations		
4	Click 'YES' button	The system displays the main menu		
5	**Check post-condition 1**			
6	Repeat steps 1,2,3 using another PIN say '6666' and click 'NO' button	The system is exited and displays a greeting message asking the user to place his ATM card in the machine		
7	**Check post-condition 2**			
8	Repeat steps 1,2, using another PIN say '7777'	The system displays a message asking the user to confirm (re-enter) the new PIN		
9	Enter a wrong confirmation (say '9876')	The system displays a message of unsuccessful operation and asks the user to confirm the correct PIN		
10	Re-enter '7777'	The system displays a message of successful operation The system asks the user if he wants to perform other operations		

What is a Test Plan?

While a test case is only designed to test a particular scenario, a test plan is a comprehensive document that lays out all major activities associated with a particular testing project. A test plan includes:

- Scope of the project
- Objectives
- Target market
- Assumptions
- Testing cycle start/end dates
- Major roles and responsibilities/overall resources
- Testing environment
- Deliverables
- Major risks and how to handle these risks
- Defect reporting and mitigation
- Testing end date

This document is designed to be a resource for both the testing teams and other teams or stakeholders.

Testing Process

1. Run the tests as defined by the test plan.

   Note : Testing should not stop when the first problem is encountered, unless it is so severe that the rest of the tests would be meaningless. Rather all testing in the test plan should be carried out and then the errors addressed.

2. Record the outcome of each test in the test report, both success and failure should be reported. For failed tests the nature of the problem should be described in sufficient detail to allow it to be corrected and to allow analysis of the types of errors being found.

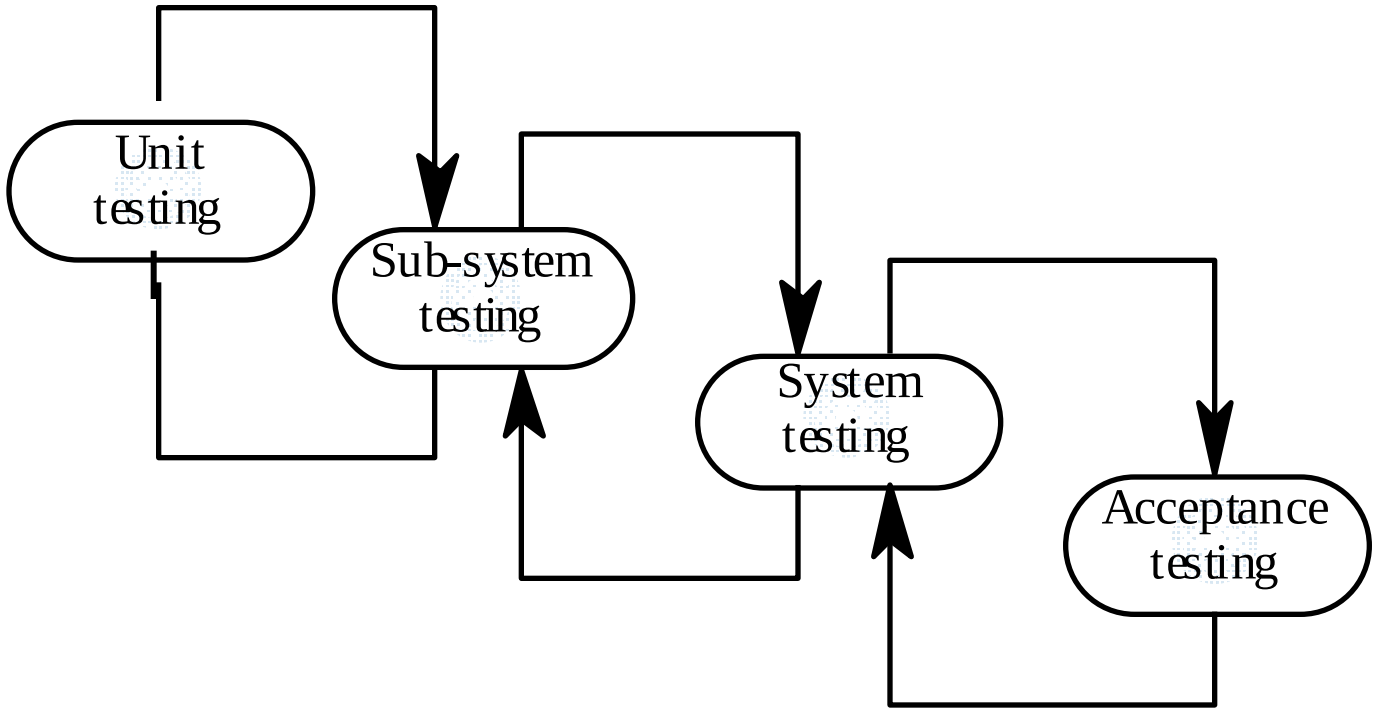
3. Correct the errors that were documented from the test run.

4. Repeat the process until no errors are identified or error rate is sufficiently low. If the error rate is low then it may be sufficient to simply re-test the failed errors. If the error rate is high then all tests should be re-run.

Dynamic and Static Verification

- *Dynamic verification* Concerned with exercising and observing product behaviour (testing)
  - includes executing the code
- *Static verification* Concerned with analysis of the static system representation to discover problems
  - does not include execution

The Dynamic Testing Process



individual components

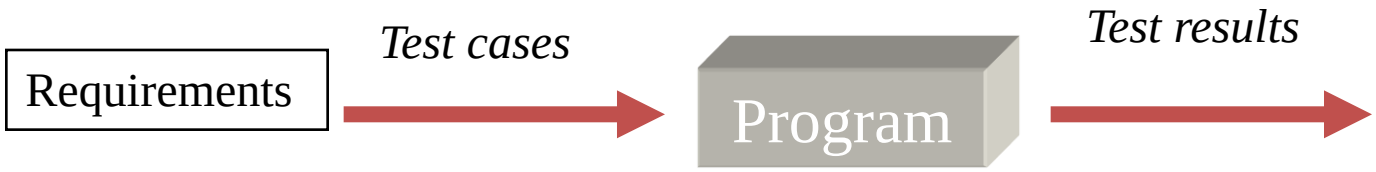
collections of components (sub-systems

The whole finished system - developers

The finished system - users

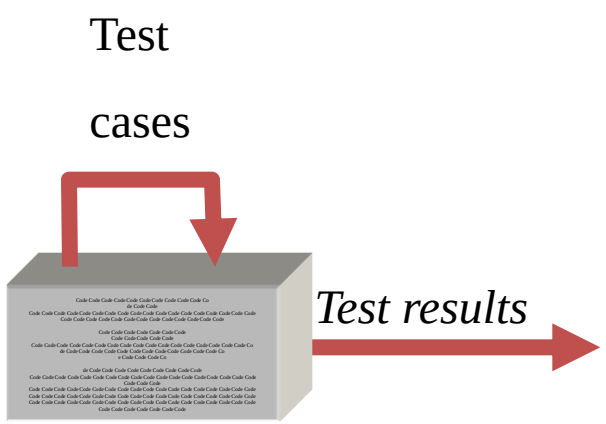
Black-Box Testing

- Approach to testing where the program is considered as a 'black-box'
- The program test cases are based on the system specification
- Inputs from test data may reveal anomalous outputs i.e. defects
- Test planning can begin early in the software process
- Main problem - selection of inputs
  - equivalence partitioning



Structural Testing

- Sometimes called white-box testing or glass box testing
- Derivation of test cases according to program structure. Knowledge of the program used to identify additional test cases
- Objective is to exercise all program statements (not all path combinations)

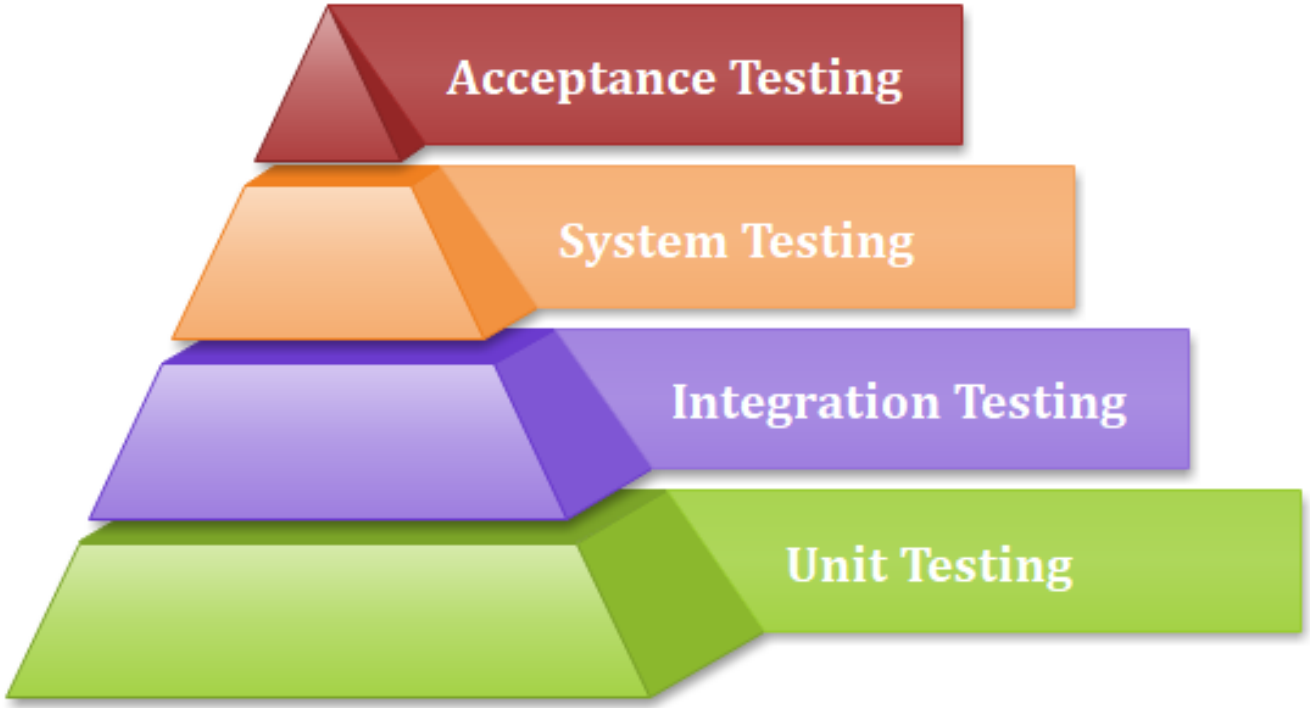


Static Verification

- Verifying the conformance of a software system and its specification ***without*** executing the code.
- Involves analysis of source text by humans or software on ANY documents produced as part of the software process
- Discovers errors early in the software process
- Usually more cost-effective than testing for defect detection at the unit and module level
- > 60% of program errors can be detected program inspections
- 2 main techniques
  - **Code walkthroughs –** The author explain the code to the other team members. They examine the code and suggest improvements.
  - **Code reviews (program inspections)** – The author distribute the code lists to the team members. At a review meeting improvements are suggested.

Test Phases

Test Phase	Test Plan	Author	Technique	Run by
Unit Test	Code design	Designer	White Box, Black box, static	Programmer
Integration Test	Functional specification	Author of specification	Black box, white box, Top-down, bottom-up	Programming team
System Test	Requirements	Analyst	Black box, stress testing,performance testing	System test team
Acceptance Test	Requirements	Analyst/customer	Black box	Analyst/customer
Alpha Test	No test plan		Black box	Selected set of users
Beta Test	No test plan		Black box	Any user
Regression Test	Functional specification/Requirements	Analyst	Black box	Development team, system test team


Acceptance Testing
System Testing
Integration Testing
Unit Testing

Unit Testing

Unit Testing is carried out as a part of the coding task. This phase is based on the design of the software for a piece of code. Unit testing should prove the following about the code

- **Robust –** the code should not fail under any circumstances.
- **Functionally correct –** the code should carry out the task defined by the code design
- **Correct interface –** the inputs and outputs from the code should be as defined in the design

Unit Test Plan –

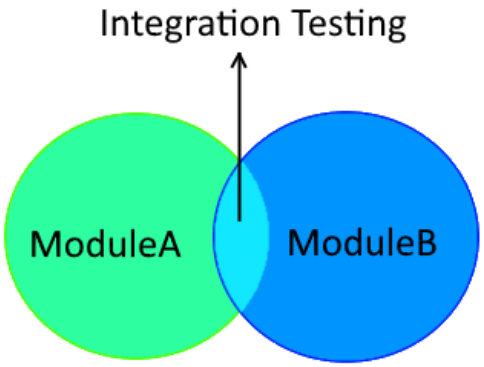
The unit test plan must be based on the design of the code and not the code itself. Therefore, the test plan will be written after the completion of design but before the start of the coding phase.

The test plan is the responsibility of the designer of the code. The testing is usually carried out by the author of the code.

Integration/Sub-systems Testing

Integration Testing is carried out after the separate software modules have been unit testing. Integration testing is based on the functional specification of the software. Integration testing should prove the following about the software:

- **Integration -** the modules of the system should interact as designed.
- **Functionally correct -** the system should behave as defined in the functional specification

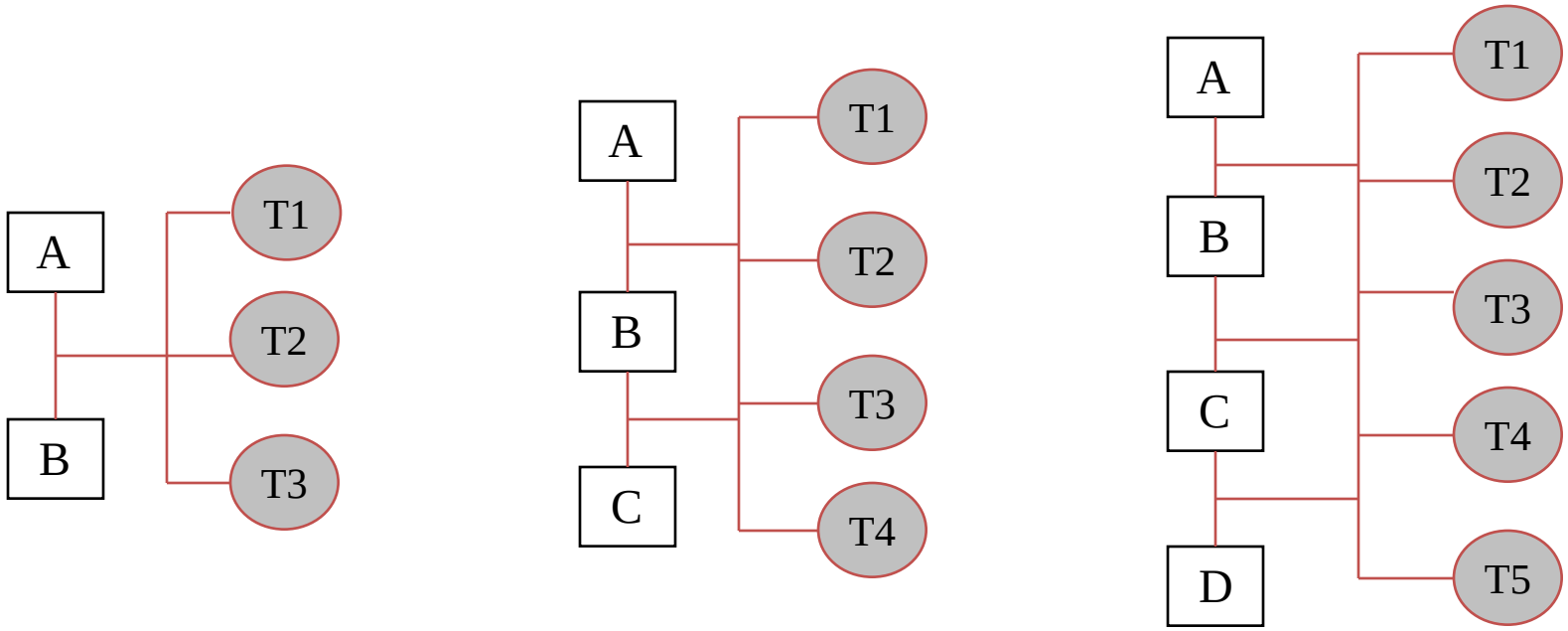


Integration test plan –

This is based on the specification of the system. The test plan is usually written by the authors of the system specification to avoid any assumptions made during design from being incorporated into the test plan.

Incremental Integration Testing

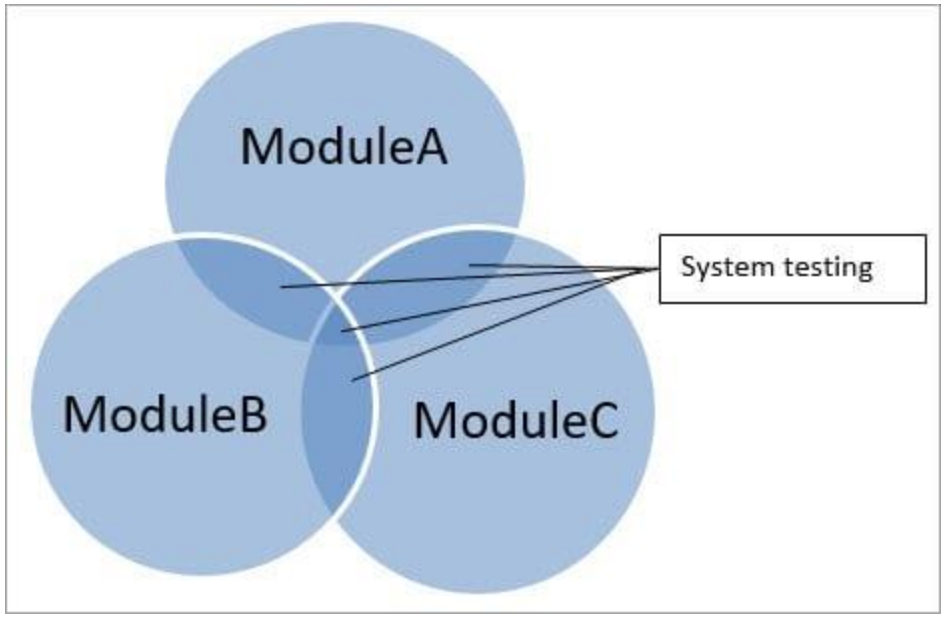
Initially, integrate a minimal system configuration and test the system. Then add components to this minimal configuration and test after each added increment.



T1,T2 and T3 tests are carried out after integration of components A and B. If the tests are successful, add C and carry out tests T1,T2,T3 and T4. If new errors introduced, that is due to the integration of C.

System Testing

System testing is carried out at the completion of the integration testing. The purpose of system testing is to prove that the software meets the agreed user requirements and works in the target environment. System testing covers both functional and non-functional requirements.



System Testing (Cont…)

System Test Plan-

The system test plan is based around the agreed requirements. Test plan covers functional requirements and non-functional requirements such as performance. The system test plan is written by the author of the requirements document to avoid assumption introduced during specification. The system test plan will also include tests to cover:

- **Recovery –** Force the system to crash and then try to recover to a sensible state.
- **Security –** Attempt to access the system without the correct authority, or attempt to carry out restricted functions.
- **Stress -** Attempt to break the system by overloading it.
- **Performance-** Ensure the system meets the performance requirements.

User Acceptance Testing

User acceptance testing is carried out at the customers site with the customer in attendance. The purpose of the acceptance test is to show to the customer that the software does indeed work. These tests are usually a subset of the system test.

Acceptance test plan –

This should be agreed with the customer after the requirements for the software have been agreed. Sometimes the customer will write the test plan in which case it must be agreed with the software developers.

Alpha, Beta and Regression Testing

Alpha Testing –

Alpha testing is the first real use of the software product. Having completed system testing the product will be given to a small number of selected customers or possible just distributed within the company itself. The software will be used and errors reported to the development and maintenance team.

Beta Testing –

After alpha testing has been completed and any changes made a new version of the product will be released to much wider audience. The objective of Beta testing is to iron out the remaining problems with the product before it is put on the general release.

Alpha, Beta and Regression Testing (Cont…)

Regression Testing –

Regression testing carried out after any changes are made to a software application. The purpose of regression test is to prove that the change has been made correctly and that the change has not introduced any new errors.

Regression test plans are usually a sub-set of the integration or systems test plans. It is designed to cover enough functionality to give confidence that the change has not effected any other part of the system.

Review Questions

1. Explain the difference between validation and verification in software engineering. Give examples of two validation and two verification activities in a typical software development process.