5.1 TESTING FUNDAMENTALS AND PRINCIPALS:

- Software testing is the process of establishing confidence that a program or system does what it is supposed to.
- Testing is the process of executing a program or system with the intent of finding error.
- Software testing can be stated as the process of validating and verifying that a computer program/application/product:
  - Meets the requirements that guided its design and development,
  - Works as expected,
  - Can be implemented with the same characteristics,
  - Satisfies the needs of stakeholders.

➢ **PURPOSE OF TESTING:**
  - To improve quality by make software error free.
  - Testing is about verifying that what was specified is what was delivered: it verifies that the product (system) meets the functional, performance, design, and implementation requirements identified in the procurement specifications.
  - The testing program is used to identify when the work has been "completed" so that the contract can be closed, the vendor paid.

➢ **ADVANTAGES:**
  - Helps to improve the quality, reliability & performance of the system.
  - Check what all functions software supposed to do & also check that Software is not doing what he not supposed to do.
  - Help to identify the defects in the early stage & try to avoid the cost of fixing the bugs.
  - Improve software quality by making software defect free.
  - To ensure that product works as user expected.

- **DISADVANTAGES:**
  - Require development time to test software internally and externally.
  - Require skill developer for white box testing
  - Increase cost of software development.
  - Require more time to release software to the customer.
  - Need various tool for white box testing.

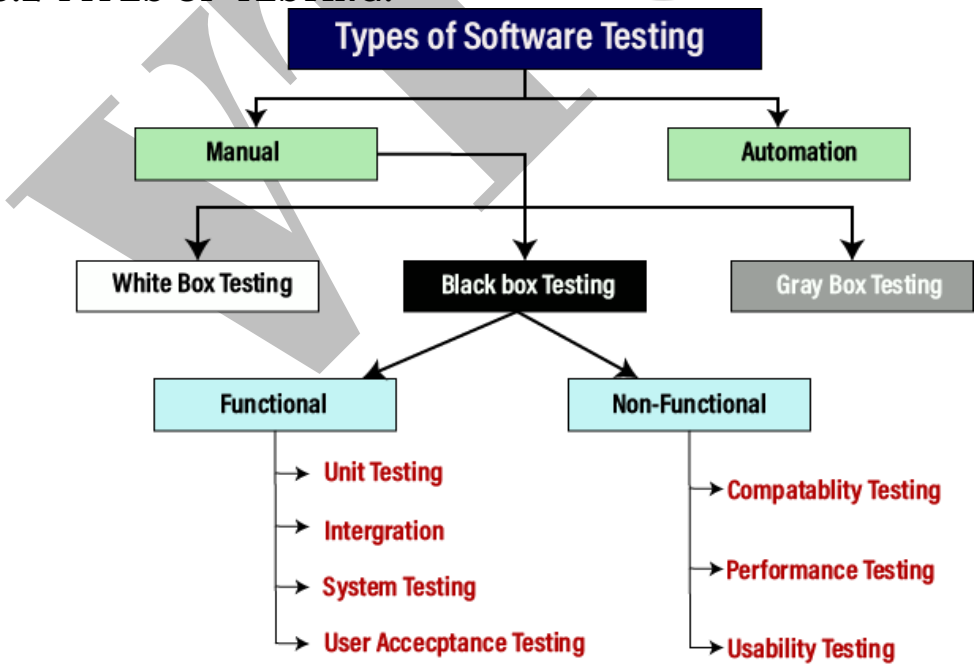
➢ ***TESTING PRINCIPLES:***

- **Testing shows presence of errors:**
  - Testing an application can only reveal that one or more defects exist in the application, however, testing alone cannot prove that the application is error free. Therefore, it is important to design test cases which find as many defects as possible. of existing, not discovered error conditions within the test object.
- **Exhaustive testing is not possible:**
  - Unless the application under test (UAT) has a very simple logical structure and limited input, it is not possible to test all possible

combinations of data and scenarios. For this reason, risk and priorities are used to concentrate on the most important aspects to test.

- **Test early and regularly:**
  - Early testing helps detecting errors at an early stage of the development process which simplifies error correction.
- **Fading effectiveness:**
  - The effectiveness of tests fades over time. If test-cases are only repeated, they do not expose new errors. Errors, remaining within untested functions may not be discovered. In order to prevent this effect, test-cases must be altered and reworked time by time.
- **Testing depends on context**
  - No two systems are the same and therefore cannot be tested the same way. Testing intensity, the definition of end criteria etc. must be defined individually for each system depending on its testing context.
- **Testing Schedule.**
  - An overall testing schedule and resource planning must be made well in advance. Hence all test can be planned and designed before any code has been generated.
- **The PARETO (80-20 Rule) Principle.**
  - This principle says that 20% of the problems lead to 80% of other problems. Hence, in order to concentrating on solving 80% of the problems rather, one can concentrate to solve 20 % of the problems which saves lot of troubles.
  - Applying Pareto principle to software testing, it would be correct to say that 80% of errors being concentrated in 20% of the developed product functionality.

5.2 TYPES OF TESTING:



5.2.1 BLACK BOX & WHITE BOX

- This method is named Black box because the software program, in the eyes of the tester, is like a black box; inside which one cannot see.
- It is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester.
- Test design techniques include:
  - Equivalence partitioning
  - Boundary Value Analysis
  - Cause Effect Graphing
- Also known as : functional testing**.**

Generic steps of black box testing:

- The black box test is based on the specification of requirements, so it is examined in the beginning.
- In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
- The fourth phase includes the execution of all test cases.
- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

Types of Black Box Testing:

Black box testing further categorizes into two parts, which are as discussed below:

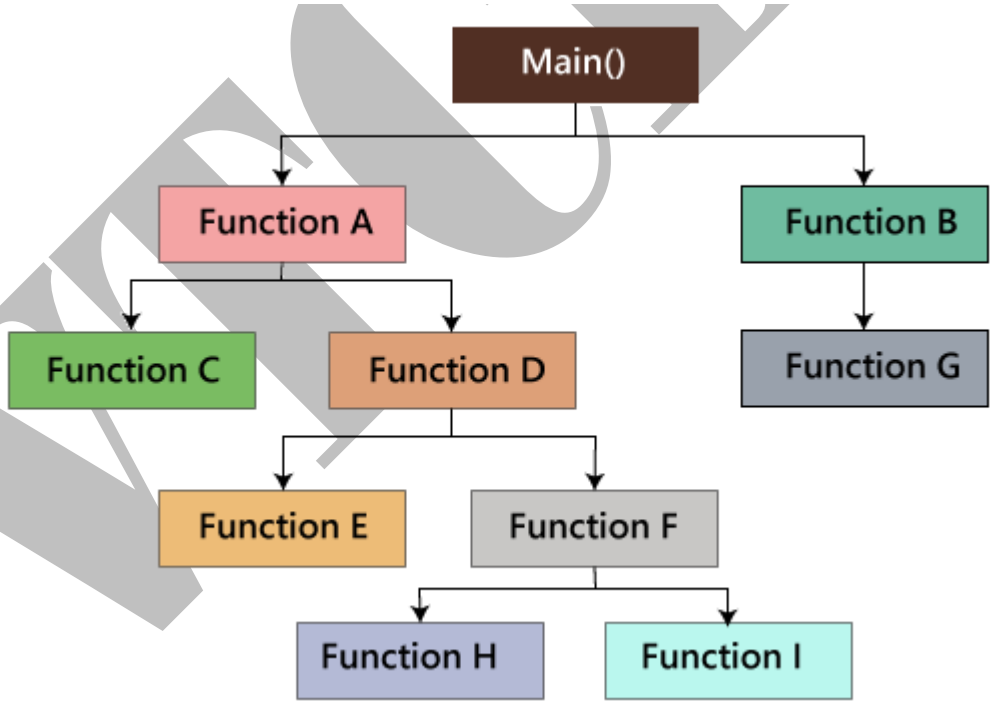
- **Functional Testing**
  - **Unit Testing**
  - **Integration Testing**
  - **System Testing**
- **Non-function Testing**
  - **Performance Testing**
  - **Usability Testing**
  - **Compatibility Testing**

➢ ***WHITE BOX TESTING:***

- It is a software testing method in which the internal structure/design/implementation of the item being tested is known to the tester.
- Also known as : clear box testing, glass box testing, structural testing.
- It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing.
- In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.
- Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer.
- The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.
- Test design techniques include:
  - **Control flow testing:** The aim of this technique is to determine the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large

program is selected by the tester to set the testing path. It is mostly used in unit testing. Test cases represented by the control graph of the program.

- **Data flow testing:** Data flow testing is used to analyze the flow of data in the program. It is the process of collecting information about how the variables flow the data in the program. It tries to obtain particular information of each particular point in the process. Such as, Variable Initialization, etc.
- **Branch testing:** Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once. Branch coverage technique is a white box testing technique that ensures that every branch of each decision point must be executed.
- **Path testing:** In the path testing, we will write the flow graphs and test all independent paths. Here writing the flow graph implies that flow graphs are representing the flow of the program and also show how every program is added with one another as we can see in the below image:



- **Code coverage testing: Code coverage** is a measure which describes the degree of which the source **code** of the program has been tested. It is one form of **white box testing** which finds the areas of the program not exercised by a set of **test** cases.

Generic steps of white box testing

- Design all test scenarios, test cases and prioritize them according to high priority number.
- This step involves the study of code at runtime to examine the resource utilization, not accessed areas of the code, time taken by various methods and operations and so on.
- In this step testing of internal subroutines takes place. Internal subroutines such as nonpublic methods, interfaces are able to handle all types of data appropriately or not.
- This step focuses on testing of control statements like loops and conditional statements to check the efficiency and accuracy for different data inputs.
- In the last step white box testing includes security testing to check all possible security loopholes by looking at how the code handles security.

Reasons for white box testing

- It identifies internal security holes.
- To check the way of input inside the code.
- Check the functionality of conditional loops.
- To test function, object, and statement at an individual level.

Advantages of White box testing

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

Disadvantages of White box testing

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

White-box testing	**Black box testing**
The developers can perform white box	The test engineers perform the black box

testing.	testing.
To perform WBT, we should have an understanding of the programming languages.	To perform BBT, there is no need to have an understanding of the programming languages.
In this, we will look into the source code and test the logic of the code.	In this, we will verify the functionality of the application based on the requirement specification.
In this, the developer should know about the internal design of the code.	In this, there is no need to know about the internal design of the code.

5.2.2 UNIT TESTING:

- It focuses on the smallest unit of software design. In this, we test an individual unit or group of interrelated units. It is often done by the programmer by using sample input and observing its corresponding outputs.
- Example:
  a) In a program we are checking if loop, method or function is working fine
  b) Misunderstood or incorrect, arithmetic precedence.
  c) Incorrect initialization
- Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.
- A unit is a single testable part of a software system and tested during the development phase of the application software.
- The purpose of unit testing is to test the correctness of isolated code. A unit component is an individual function or code of the application. White box testing approach used for unit testing and usually done by the developers.
- Whenever the application is ready and given to the Test engineer, he/she will start checking every component of the module or module of the application independently or one by one, and this process is known as **Unit testing** or **components testing**.
- Unit testing helps tester and developers to understand the base of code that makes them able to change defect causing code quickly.
- Unit testing helps in the documentation.
- Unit testing fixes defects very early in the development phase that's why there is a possibility to occur a smaller number of defects in upcoming testing levels.
- It helps with code reusability by migrating code and test cases.

5.2.3 INTEGRATION TESTING:

- The objective is to take unit tested components and build a program structure that has been dictated by design. Integration testing is testing in which a group of components is combined to produce output.
- Integration testing is of four types:

■ **Top-down:**

Top-down testing is a type of incremental integration testing approach in which testing is done by integrating or joining two or more modules by moving down from top to bottom through control flow of architecture structure. In these, high-level modules are tested first, and then low-level modules are tested. Then, finally, integration is done to ensure that system is working properly. Stubs and drivers are used to carry out this project. This technique is used to increase or stimulate behavior of Modules that are not integrated into a lower level.

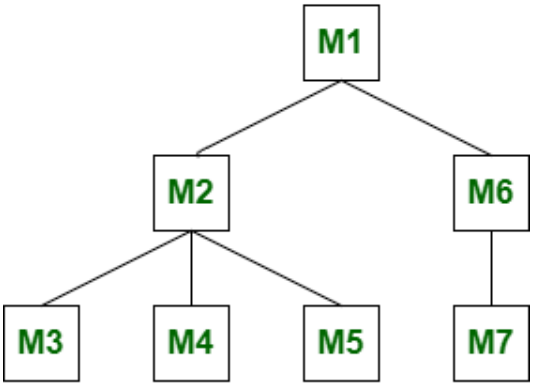
Processing :

Following are the steps that are needed to be followed during processing :

1. Test driver represents main control module also known as a high-level module and stubs are generally used for all low-level modules that directly subordinate (present or rank below another) to high-level modules.
2. In this, testing takes place from to bottom. So, too or high-level modules are tested first in isolation.
3. After this, low-level modules or subordinated modules or stubs replace high-level module one by one at a time. This can be done using methods like depth-first or breadth search.
4. The process is repeated until each module is integrated and tested.
5. Another stub replaces present real or control module after completion of each set of tests. These stubs act as s temporary replacement for a called module (stubs) and give same result or output as actual product gives.
6. To check if there is any defect or any error occurred or present, regression testing is done and it's important to reduce any side effect that might be caused due to errors occurred.

Example –

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.



Program Structure

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.

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.

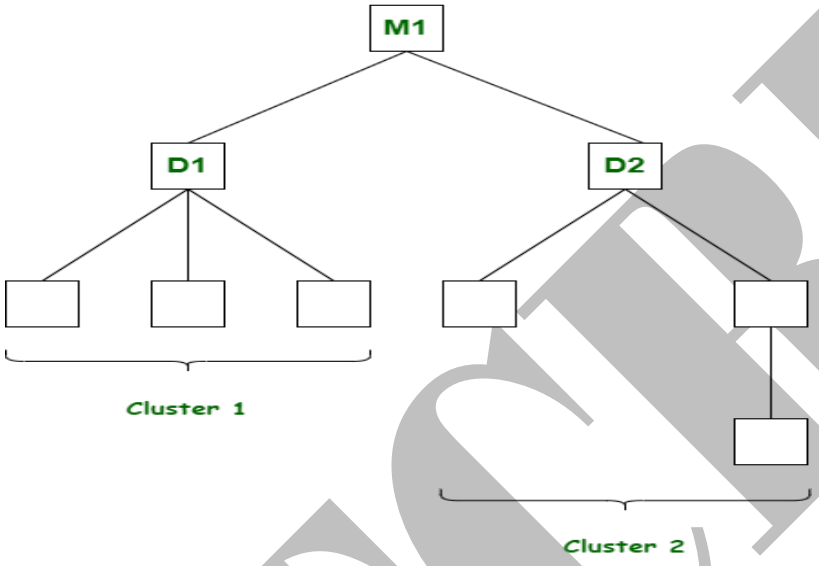
Disadvantages :

- 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:**

- **Bottom-up Testing** is a type of incremental integration testing approach in which testing is done by integrating or joining two or more modules by moving upward from bottom to top through control flow of architecture structure. In these, low-level modules are tested first, and then high-level modules are tested.
- This type of testing or approach is also known as inductive reasoning and is used as a synthesis synonym in many cases. Bottom-up testing is user-friendly testing and results in an increase in overall software development. This testing results in high success rates with long-lasting results.
- **Processing :**
  Following are the steps that are needed to be followed during the processing :

1. Clusters are formed by merging or combining low-level modules or elements. These clusters are also known as builds that are responsible for performing the certain secondary or subsidiary function of a software.
2. It is important to write a control program for testing. These control programs are also known as drivers or high-level modules. It simply coordinates input and output of a test case.
3. Testing is done of entire build or cluster containing low-level modules.
4. At lastly, control program or drivers or high levels modules are removed and clusters are integrated by moving upward from bottom to top in program structure with help of control flow.



Bottom Up Integration Testing

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 :

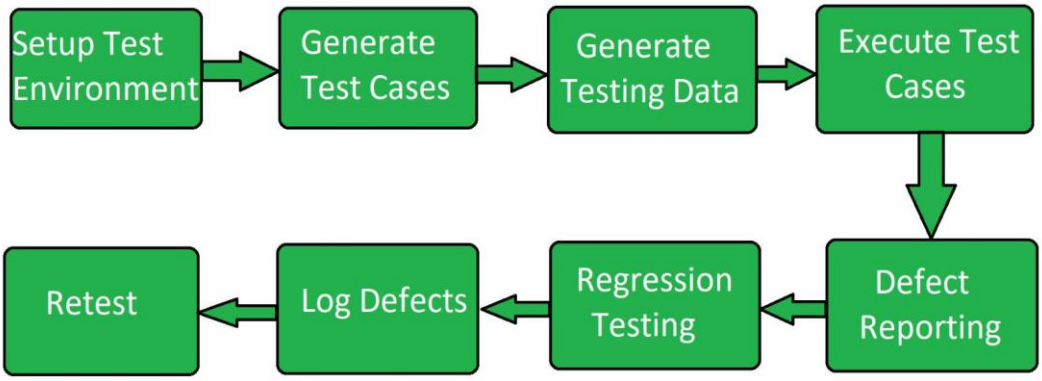
- Towards top of the Hierarchy, it becomes very complicated.
- There is no concept regarding early skeletal system.
- There will be an impact on sibling and higher-level unit tests due to changes.

5.2.4 SYSTEM TESTING

- This software is tested such that it works fine for the different operating systems. It is covered under the black box testing technique. In this, we just focus on the required input and output without focusing on internal working.
- In this, we have security testing, recovery testing, stress testing, and performance testing .
- **System Testing** is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in

the context of both. System testing tests the design and behavior of the system and also the expectations of the customer.

- Example:
  - This include functional as well as non functional testing
- **System Testing Process:**
  System Testing is performed in the following steps:
  - **Test Environment Setup:**
    Create testing environment for the better quality testing.
  - **Create Test Case:**
    Generate test case for the testing process.
  - **Create Test Data:**
    Generate the data that is to be tested.
  - **Execute Test Case:**
    After the generation of the test case and the test data, test cases are executed.
  - **Defect Reporting:**
    Defects in the system are detected.
  - **Regression Testing:**
    It is carried out to test the side effects of the testing process.
  - **Log Defects:**
    Defects are fixed in this step.
  - **Retest:**
    If the test is not successful then again test is performed.



Types of System Testing:

- **Performance Testing:**
  Performance Testing is a type of software testing that is carried out to test the speed, scalability, stability and reliability of the software product or application.
- **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.
- **Stress Testing:**
  Stress Testing is a type of software testing performed to check the robustness of the system under the varying loads.

- **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.

5.3 INTRODUCTION TO CHANGE OVER

- As technologies change, many businesses find themselves needing to change over their computer information systems.
- Upgrading these systems helps them optimize their efficiency and remain competitive.
- Common changeover areas include security systems, database systems, accounting systems and managerial information systems.
- Deciding which changeover technique will work best for a particular company depends on the type of changeover and degree of risk for the company**.**

5.3.1 TYPES OF CHANGE OVER

1.Data Conversion / Change over:

- It is a Important part of system installation process.
- During data conversion, existing data is loaded into new system.
- It can be done before, after or during operational environment is complete.
- We should develop data conversion plan as early as possible and the conversion process should be tested when tested environment is developed.
- When new system replace an existing system, you should automate the data conversion process.
- The old system might be capable of exporting data in an acceptable format of the new system.
- If a standard format is not available, you must develop a program to extract the data and convert it into acceptable format.
- Data conversion is more difficult when the new system replace manual system because all data must be entered manually.

2. Application / System Changeover:

- Definition: Changeover or Conversion is the process of changing from the old system, which is currently running in the organization, to the newly built system.
- There are four methods of handling the system conversion which are :-
  - ☐ Parallel system method.
  - ☐ Dual system method.
  - ☐ Direct cutover method.
  - ☐ Pilot Approach method.

3. Parallel Changeover:

- It require both old and new system operate fully for a specified period.
- Data is input in both system and output generated by new system is compared with the equivalent output from the old system.

- When user, management and IT group are satisfied that the new system operates correctly , old system is terminated.

Advantages:

- Lower risk: If the new system does not work correctly, company can use old system as backup .
- Easier to verify new system: Output of new system is compare with old system and verified during parallel operation.

Disadvantages

- Costly: Company pay for both system because both system are in full operation.
- Processing Delay: Running both system might place a burden on operating environment and cause processing delay.
- Work and data are duplicated.
- Time consuming.

4.Direct Changeover:

- Direct changeover, also referred to as immediate replacement, tends to be the least favorite of the changeover techniques.
- In a direct changeover, the entire system is replaced in an instant.
- This involves taking the old system offline and putting the new system online within a day or over a weekend or holiday period.
- It is least expensive because IT group has to operate and maintain only one system at a time.
- There are no parallel activities and there is on falling back to old system.
- It is only choice if the operating environment cannot support both the old and new system or if the old and new system are incompatible.
- Most organization use direct cutover only for noncritical situations.
- Timing is important when using direct changeover strategy.

Advantages

- Minimal Cost : IT group has to operate and maintain only one system at a time.
- Minimize Workload.

Disadvantage

- More Risk : Because if Changeover fail stop entire organization.
- Require careful planning: If something goes wrong, reverting back to the old system usually is impossible.

5. Phase In / Dual Changeover:

- The phased changeover technique is considered a compromise between parallel and direct changeovers.
- In a phased changeover, the new system is implemented one stage at a time.
- Phased operation works in different phases or stages.
- Implementation of new system in modules or stages is phased operation.
- In this method, new system is implemented in many phases over a period of time. So, the old system is gradually phased out while the new one is being phased in.
- In phase operation the risk of errors or failures is limited to the implemented module only and also phased operation is less expensive than the full parallel operation.

- The actually conversion from the old parts of the system to the new parts may be either parallel or direct.
- As an example, consider a company working toward installing a new financial system. Implementing the new system one department at a time, the company converts accounts receivable, accounts payable, payroll, and so on .
- This method is used when it is not possible to install a system throughout an organization all at once.

ADVANTAGE

- Less risky because implemented in phase so no danger of total breakdown.
- Any problem should be in one area other area operations are unaffected.

DISADVANTAGE

- Can take long time to achieve total changeover.
- Interface between parts of the system may make this impractical.

6. Pilot Changeover:

- In the Pilot Approach method, a complete new version of the system is implemented in just one location or Site of the organization.
- For example, a bank may first test the system at one of its branches. This branch is referred to as the pilot, or beta, site for the program.
- The old system continues to operate for the entire organization including the pilot site.
- After the system proves successful at the pilot site, it is implemented in the rest of the organization, usually using direct cutover method.
- Pilot operation is combination of parallel operation and direct cutover methods.

ADVANTAGE:

- Less risky
- Less costly

DISADVANTAGE

- Can take a long time to achieve total changeover.