System Engineering and Quality Assurance

Design Objectives

- ✓ The two operation design objective continually sought by developer is system reliability and maintainability.
- ✓ For designing RELIABLE system the following objectives are important.

1 Design Reliable system

2 Approaches to Reliability

- Error avoidance
- Error detection and correction
- Error tolerance

Design Objectives

1) Designing reliable system:-

- A systemic said to have reliability if does not produce dangerous or costly failures when it is used in a reasonable manner, that is in a manner that a typical user expect is normal.
- This definition recognizes that system may not always be used in the ways that design expect
- However there are step analysts can take to ensure that the system is reliable when it is installed and that the reliability can be maintain after implementation

2) Approaches to reliability:-

- There are two approaches of reliability.
- The first is that the system is meeting the right requirements.
- For instance a system a might be expected to have specific security features or built into it by the users.
- The second level of system reliability involves the actual working of the system Delivered to the user
- At this level, system reliability is with software engineering and development.
- An error occurs whenever the system does not produce the expected output.
- While it is true that no program is ever fully debugged or fully tested, nor proven correct –a fact that startles many user and aspiring programmer-error are not limited to the correct use of programming syntax alone.

Approaches to reliability		
Approach	**Description**	**Example**
Error avoidance	Prevents error occurring in the software	Is impossible in large system
Error detection and correction	Recognizes error when they encountered and corrects the error or the effect of the error so the system does not fail	Tarps and modified illegal arithmetic step unexpected data values.
Error tolerance	Recognizes error when they occur ,but enable the system to keep running through degraded performance or by applying rules that instruct the system how to continue processing	Shutdown part of system does not perform some processing but keeps the system the operational.

➢ Error avoidance :-

- There are three approaches to reliability in table
- Under error avoidance, developers and programmers make every attempt to prevent error from occurring at all.
- Analyst must assume that it is impossible to fully achieve this objective
- Error will occur despite the best the efforts of every competent people.

➢ Error detection and correction:-

- This method uses design future that detects error and makes necessary changes to correct white the program is in use or the effect on the user, so that a failure does not occur.

➢ Error tolerance:-

- Error tolerance strategies keep the system running even in the presence of error.
- Another manner of error tolerance is the use of degraded processing.
- With this strategy, the use receive less service than the system was designed to provide ,but that is considered a better alternative in some cases than having no service at all.

Quality Assurance

- Quality Assurance is the review of s/v products & related documentation for completeness, correctness, reliability & maintainability.
- It includes assurance that the system meets the specifications & the requirements for its intended use & performance.

Analysts use four levels of Quality Assurance:

1) Testing
2) Verification
3) Validation
4) Certification

(1) Testing:-

- System testing is an expansive but critical process that can take as much as 50% of the budget for program development.
- The common view of testing held by users is that it performed to prove that there are no errors in a program.
- The tester, who may be an analyst, programmer or specialist trained in software testing.

(2) Verification:-

- Have we building the software right? (i.e..does it match the specification)
- Verification is the process of evaluating a system or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

(3) Validation:-

- Have we built the right software? (i.e., is this what the customer wants)
- Validation is the process of evaluating a system or component during or at the end of the development process to determine whether it satisfies specified requirements.

(4) Certification:-

- Product certification or product qualification is the process of certifying that a certain product has passed performance and quality assurance tests or qualification requirements stipulated in regulations such as a building code and nationally accredited test standards, or that it complies with a set of regulations governing quality and minimum performance requirements.
- Certification of product may indicate their established suitability for a specified purpose (e.g. a computer system might be certified as being fully compatible with a large software package).

Software Design and Documentation Tools

Structure flowchart

- Structure chart also called nassi scheideman chart.
- This are graphics tools that force to the designer for the to draw structure chart of the system.
- They provide structure that can be retained by programmer who develop the application software.
- In some organization analyst are responsible for developing module logic or it helps to programmer when design the system.
- If there are complex design methodology for implementation large system it will require less code and appropriate level.

Basic Block

- The basic elements of a flowchart are shown in figure-1
  - The start block:-

    i. The start block represent the beginning of a process
    ii. It always has exactly one output.
    iii. The start block is labeled with a brief description of the process carried out by proceeding flowchart.

  - The end block:-

    I. The end block represents the end of a process.
    II. It always has exactly one input and generally contains either end or return depending on its function in the overall process of the flowchart.

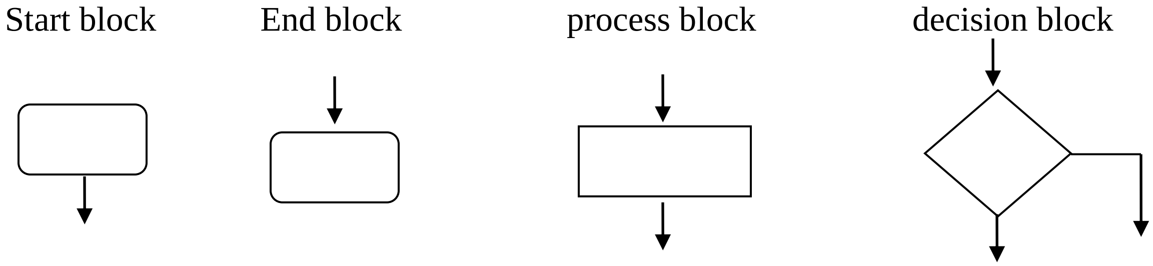


Figure 1: Basic flow chart Block

➢ **A process block:-**

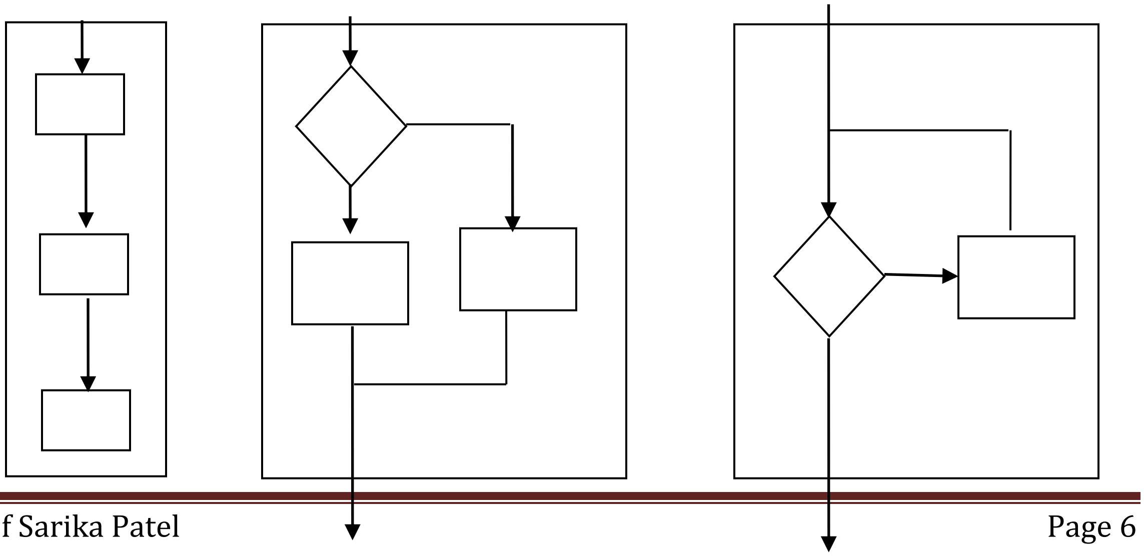
- A process block represents some operation carried out on an element of data.
- It contain a brief descriptive label describing the process being carried out on the data.
- It may itself be further broken down into simpler step by another complete flowchart representing that process.
- If it is broken down further ,the flowchart that represent the process will have the same label in the start block at the higher level.
- A process always has exactly one input and output.

➢ **A decision block:-**

- A decision block always makes a binary choice.
- The label in decision should be a question that clearly has only two possible answers.
- The decision block will have exactly one input and two outputs.
- The two output will be labeled with the two answers to the question in order to show the direction of the logic flow depending upon the decision made.

✓ **Basic structures**

- A structure flowchart id one in which all of the processes and decision must fit into one of a few basic structure elements.
- The basic element of a structure flowchart are shown in figure-2 it should be possible to take any structure flowchart and enclose all of the blocks within one of the following process structures.
- Note that each of the structure shown below has one input and one output. Thus the structure itself represented by a single process block.



Sequence structure If-Then-Else structure While structure

Figure 2: Basic Flow chart Structure

The sequence process is just a series of processes carried out one after the other .most programs are represented at the highest level sequence. possible with a loop from the end back to the beginning.

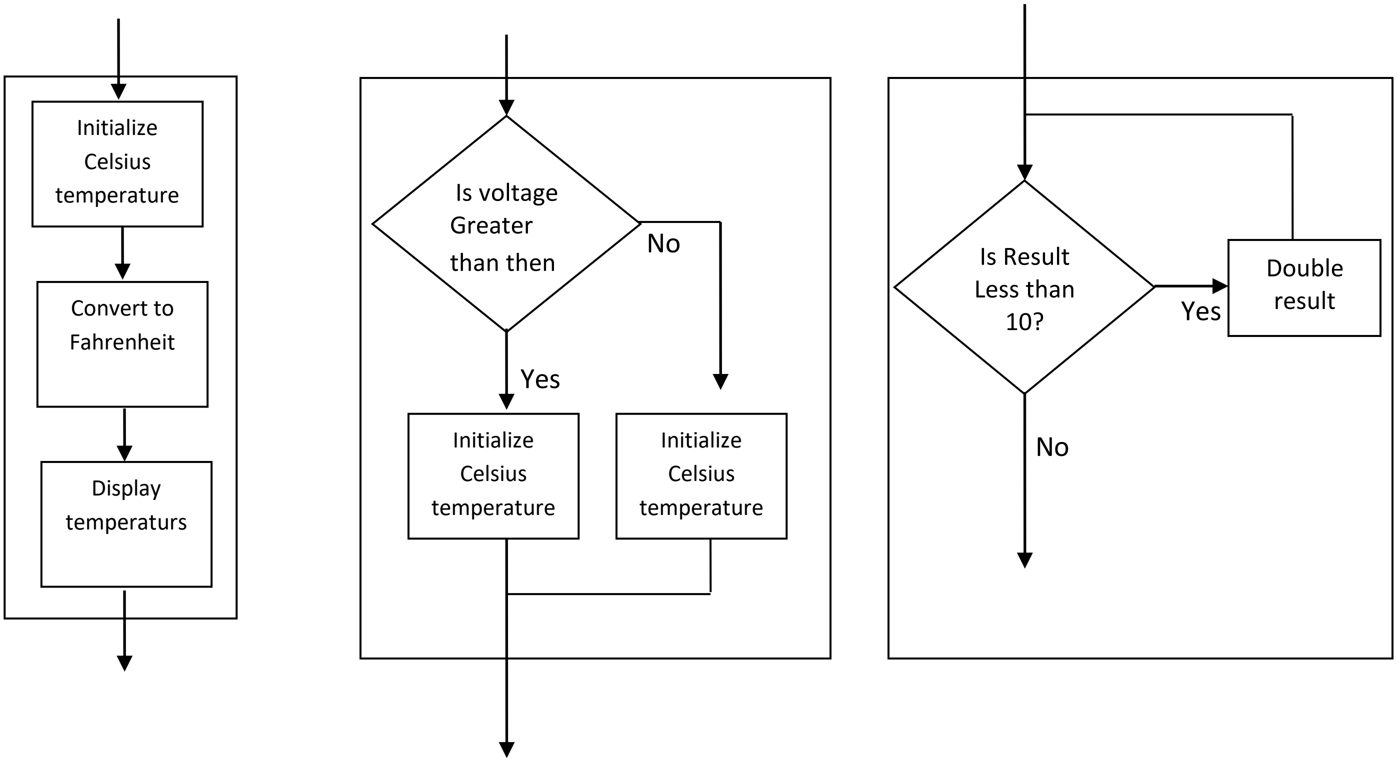
The if-then-else process logically completes the binary decision block by providing two separate processes. one of the processes will be carried out in the each path from the binary decision.

The while process allow for the representation of a condition loop structure within a program .the decision to execute the process in the loop is made the first execution of the process.

Structure Flow Chart Example

Flowchart implementation using c

Writing c code from a structure flow chart is very straightforward. Each type of process structure describe previously has a corresponding c code program flow statement figure-3 illustrate the c code implementation of the three basic and two derived process structure .note that the single process block can be implemented with a single line of code, multiple lines of code a function call or structure process block.


Initialize Celsius temperature
Convert to Fahrenheit
Display temperaturs
Is voltage Greater than then
No
Yes
Initialize Celsius temperature
Initialize Celsius temperature
Is Result Less than 10?
Yes
Double result
No

```
X = 10;

Y = 9*x / 5 + 32;

DispTemp (x, y);

                 ………….
```

```
If (x  >  5 )
{
X  =  5;
}
Else

{
X=  0;

}
```

```
While ( x =  <  10 )

{

X  =  2*xi;

}
```

Figure 3:C code Implementation for Basic Flow chart Structures

HIPO Diagram (Hierarchical Input process Output)

- Hipo is commonly used method for developing system s/w
- They are used in documenting information.

An acronym for Hierarchical input process output this method was developed by IBM for its large complex operating system.

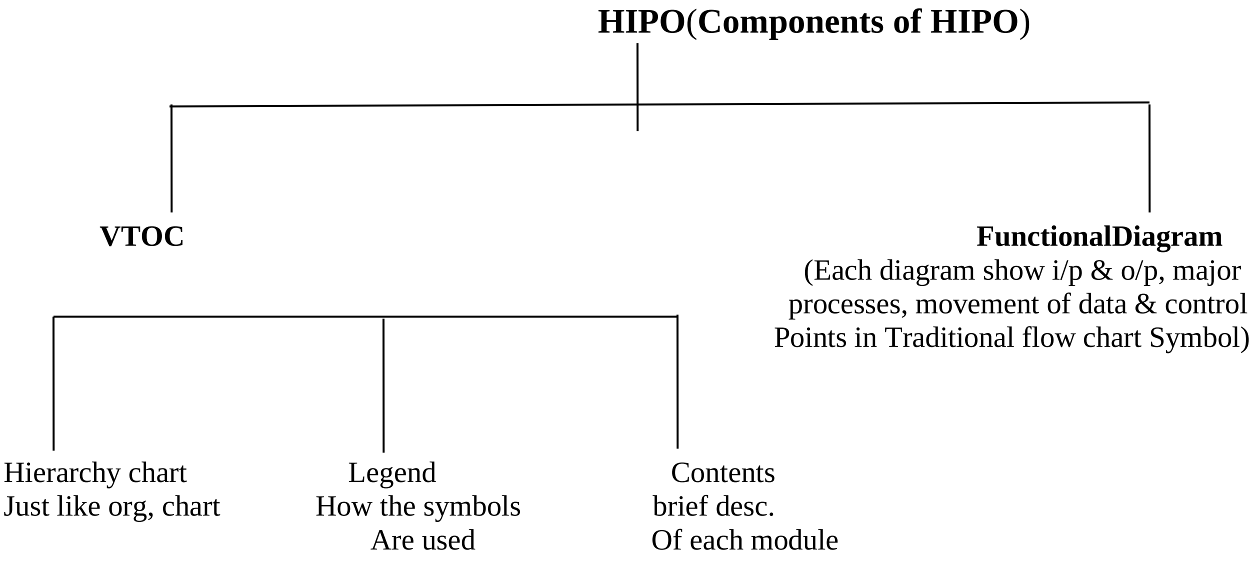
- HIPO emphasizes on Function of tine system rather than structure , logic or organization .
- HIPO diagram are graphics, rather than narrative description of the system.

They assist the analyst in answering three guideline question :

1. What does the system or module do? (Asked when designing the system)
2. How does it is do it ?(Asked when reviewing the code for testing or maintenance)
3. What are the input and outputs? (Asked when reviewing the for testing or maintenance)

A HIPO description for a system consists of

I. Visual table of content(VTOC)
II. Function Diagram



HIPO(**Components of HIPO**)

VTOC

FunctionalDiagram
(Each diagram show i/p & o/p, major processes, movement of data & control Points in Traditional flow chart Symbol)

Hierarchy chart
Just like org, chart

Legend
How the symbols
Are used

Contents
brief desc.
Of each module

❖ Advantages of Hipo

- Hipo allows a program or a system to be easily understood.
- Hipo is design, Development & documentation tool.
- HIPO package provide a common base for education & communication.

- HIPO has less duplication on information & more information can be obtained in a glance.
- It is a top –down approach with successive level going for greater details.
- in HIPO , error can be detected & isolated on a functional Basis.
- HIPO indicate clearly what input; processing & output are involved in each programming module.
- HIPO design computer programs in module that can be individually assigned. Designed & tested.
- HIPO is a management tools because the functional approach allows planning & scheduling to be made accurately & early in the system development life cycle.
- HIPO aids designers & force them to think about how specifications will be met & where activities & component must be linked together .

❖ Disadvantage of Hipo

- HIPO concentrates more on what is to be accomplished rather than how it is to be accomplished.
- Requires extra documentation beyond flow chart or coding.
- Necessitates some training for programmer before use.
- HIPO diagram are not as easy to use for communication purposes as many people would like.
- They do not guarantee error-free system.

A HIPO description for a system consists of contents and the functional diagram.

(1) Visual table of contents(vtoc)

- The visual of table of content (vtoc) show the relation between each of the documents up a HIPO package.
- It consists of the a hierarchy chart that identifies the module in a system by number & in relation to each other & gives brief description of each module.
- The module are in increasing detail.
- Depending on the complexity of the system. three to five level of module are typical.

Hierarchy chart of module:-

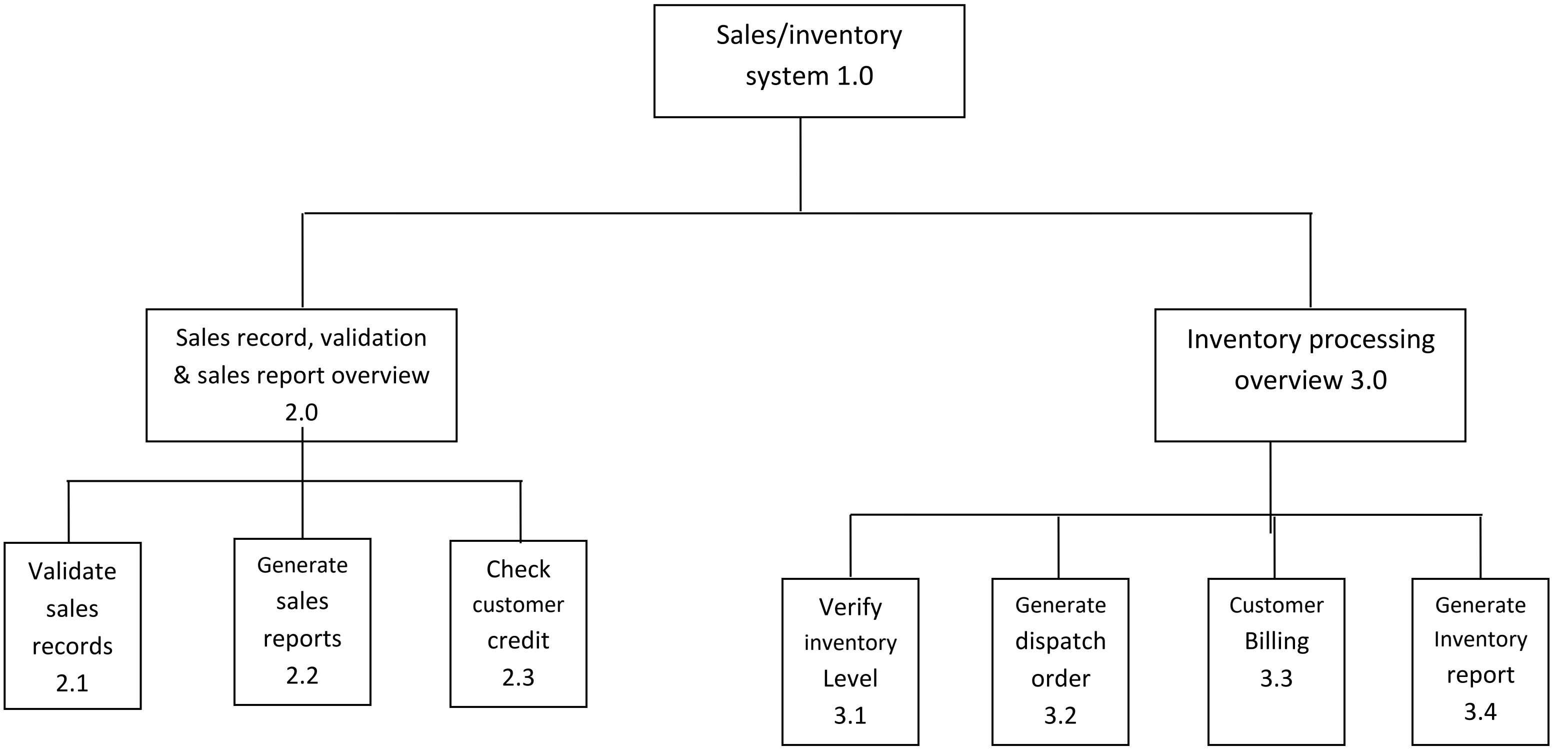

Sales/inventory system 1.0
Sales record, validation & sales report overview 2.0
Inventory processing overview 3.0
Validate sales records 2.1
Generate sales reports 2.2
Check customer credit 2.3
Verify inventory Level 3.1
Generate dispatch order 3.2
Customer Billing 3.3
Generate Inventory report 3.4

Figure-1 Hierarchy chart

Legend:-legend in VTOC tell how the symbol are used in it

⇦	**Data movement**
⇨	**Control Flow**
○⇦	**Keyed data arrow**
⇨○	**Keyed data arrow**

Figure-2 Legends

➢ Content Giving brief description of each Module.

This is optional.

1.0 Sales/ inventory system:-

- Control all processing .invokes program to handle data entry, validate record. generate report & Performs customer billing

2.0 Validate sales record:-

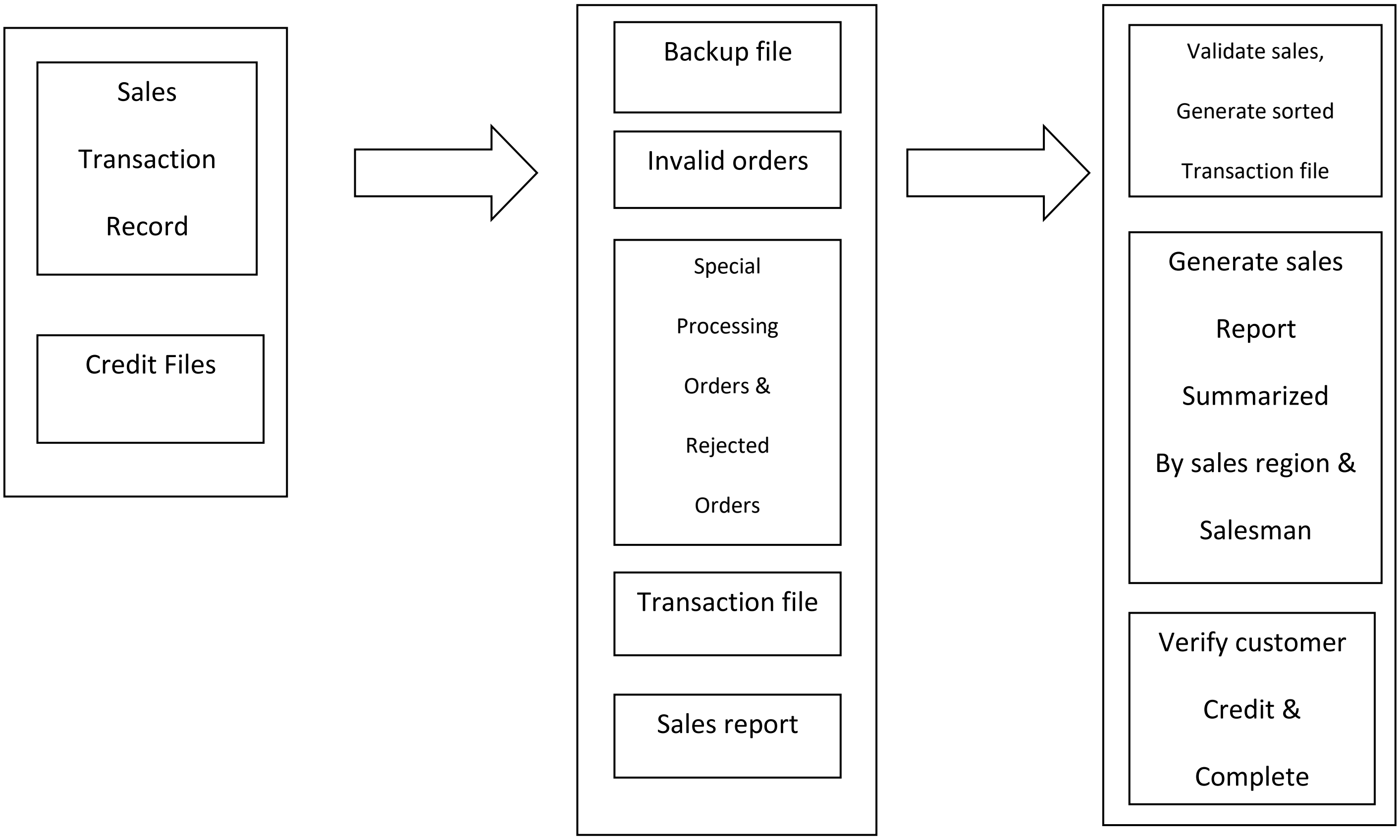
- Makes data entry check sales record check customer credit & generate sales reports.

(2) Functional diagram:-

- There is one diagram for each box in the VTOC.
- Each diagram show i/p & o/p (Right to left or Top to bottom), major processes, movement of data & control points.
- Traditional Flowchart symbol represent media, such as magnetic tape, magnetic disk & printed output.
- A solid arrow show control paths & an open arrow identifies data flow.
- Some functional diagrams contain other intermediate diagrams.
-
- They show external data as well as internally developed data & the step in the procedure where the data are used.

➢ **IPO Chart:-**

- Each functional in a HIPO chart is further described through IPO (input process output) chart. Now IPO chart contain three boxes showing the input. Process and output.

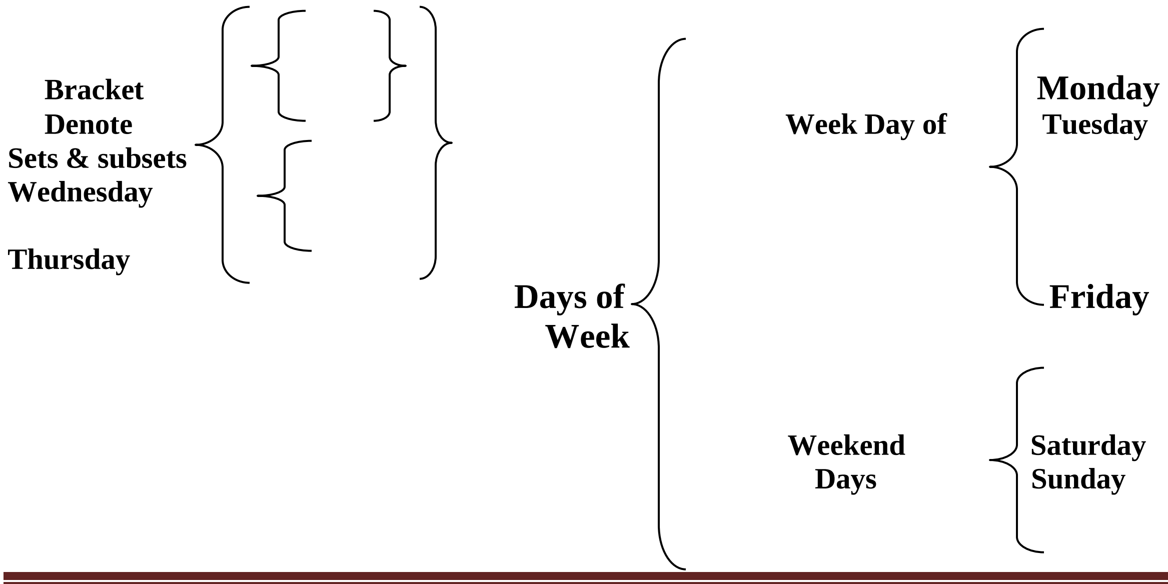

Sales Transaction Record
Credit Files
Backup file
Invalid orders
Special Processing Orders & Rejected Orders
Transaction file
Sales report
Validate sales, Generate sorted Transaction file
Generate sales Report Summarized By sales region & Salesman
Verify customer Credit & Complete

Warnier/Orr Diagram

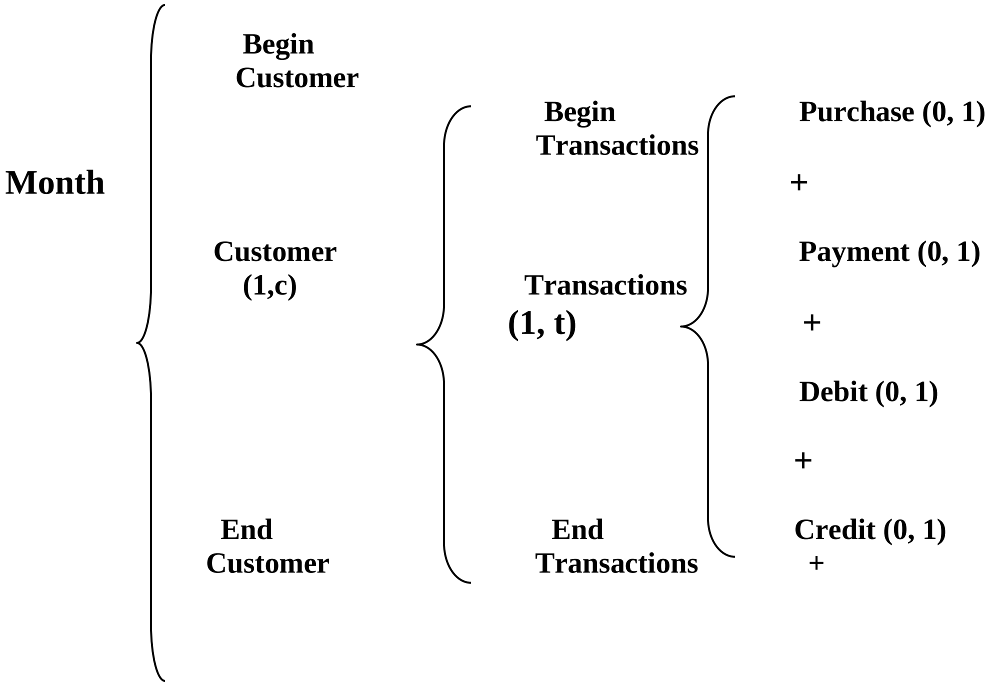
- Warnier/Orr Diagrams also known as Logical Construction of Program or Logical Construction of systems.
- Warnier/ Orr diagrams were initially developed in France by Jean Dominique Warnier & in the United states by Kenneth Orr.
- Warnier/Orr diagram technique is a system design & program design technique.
- Its approach is a structured, hierarchical method using brackets that how from left to right.
- Like all structured design methodologies, the basic building are:
- 1) Process 2)Decision 3)Iteration
- This method aids the design of program structures by identifying the output & processing result &then working backwards to determine the steps & combinations of input needed to produce them.
- The simple graphic methods used in Warnier / Orr diagrams make the levels in the system & the movement of the data between them.

Basic Elements

- Warnier/Orr diagrams show the processes & sequences in which they are performed.
- Each process is defined in a hierarchical manner: it consists of sets sub processes that define it.
- At each level, the process is shown in bracket that groups its components.



- The sets of days in the week have weekdays & weekend as subsets.
- Since a process can have many different sub-processes, a Warnier / Orr diagram uses a set of brackets to show each level of the system.
- For example, in developing an invoicing system, there are processes that occur for each transaction, each day & each month. As shown in below figure:



Using Warnier / Orr diagram

- To develop Warnier / Orr diagram, the analyst works backwards, starting with systems output & using an output-oriented analysis.
- On paper, the development moves from left to right.
- First, the intended output or results of the processing are defined.
- At the next level, shown by inclusion with a bracket, the steps needed to produce the output are defined.
- Each step in turn is further defined.
- Additional brackets group the processes required to produce the result on the next level.

Warnier /orr diagram advantages

- Warnier / Orr diagrams are simple in appearance & easy to understand.
- They are powerful design tools.
- They are showing grouping of processes & data that must be passed from level to level.
- The sequence of working backwards ensures that the system will be result oriented.
- This method is useful for both data & process definition.

System Key Concepts

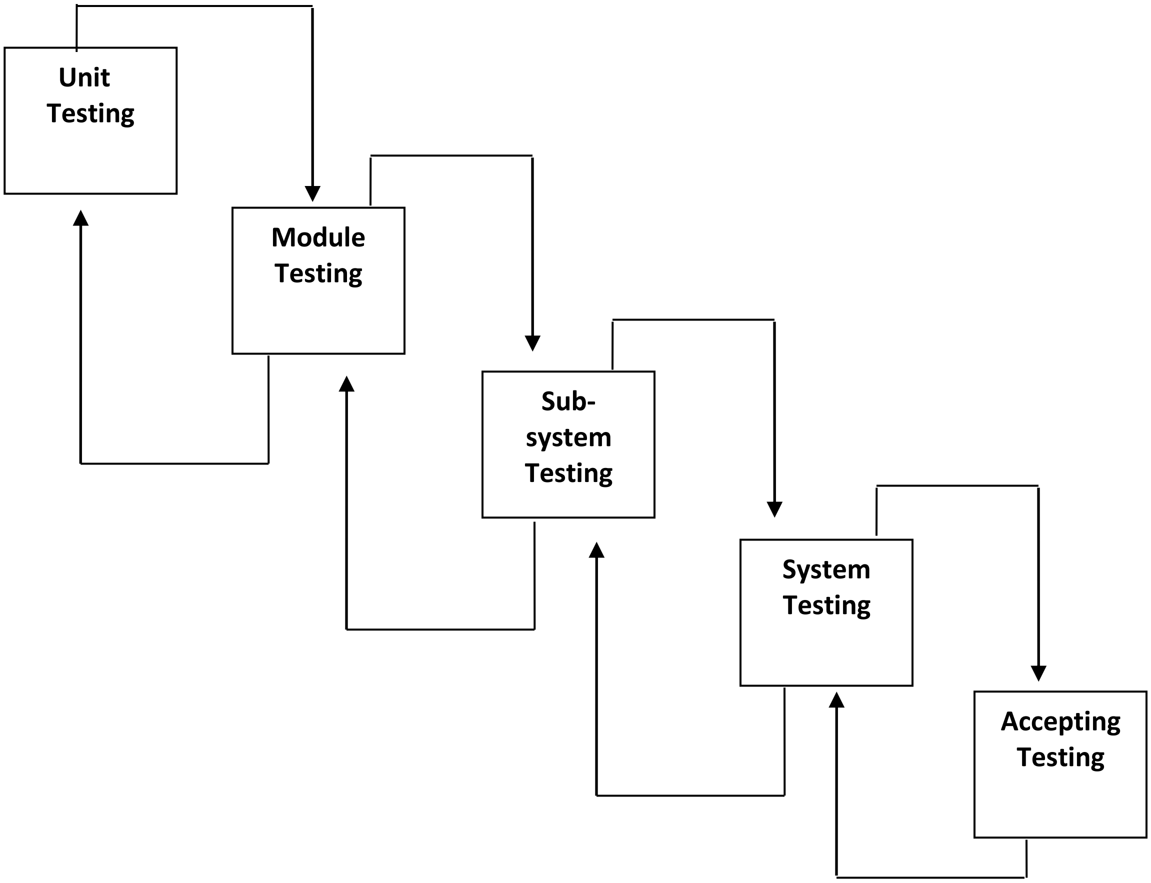
Testing principles

- A software engineer must understand these basic principles that guide software testing.

1) All tests should be traceable to customer requirements. It follows that the most severe defects from the customer point of view are those that cause the program to fail to meet its requirements.
2) Testing schedule:- An overall testing schedule and resource planning must be made well in advance. Hence all tests can be planned and designed before any code has been generated, that is immediately after the requirements model is complete.
3) The praetor (reduce) principal:- The principal implies 80% of all errors uncovered during testing will be likely be traceable to 20 %of all program modules. Hence the problem is to isolate these suspect modules and to thoroughly test them.
4) Testing should begin:- “In the small” and business towards testing “in the large” . the first best planned and executed generally focus on the individual program modules. As testing progresses, testing shifts focus in an attempt to find errors in integrated clusters of modules and ultimately in the entire system.
5) Exhaustive (complete) testing is not possible for any system.
6) To be most effective, testing should be conducted by independent third party.

The testing process

- Expect for small programs, systems should not be tested as an single unit.
- Large system is built out of subsystems which are built out of modules which in turn composed of procedures and functions.
- The testing process should therefore proceed in stages where testing is carried out incrementally in conjunction with system implementation.
- The most widely used testing process consists of five stages as shown in the fig. below.



- The process is an iterative one with information begins feedback from later stages to earlier parts of the process.
- The stages in the testing process are.

  Unit testing
  Module Testing
  Sub-System Testing
  System Testing
  Accepting Testing

(1) Unit Testing:-

Individual components are tested to ensure that they operate correctly. Each component is tested independent components.

(2) Module Testing:-

A module testing is a collection of dependent component such as an object class, an abstract data type or some looser collection or procedures and function.
A module encapsulates related components so that it can be tested without other system modules.

(3) Sub-System Testing:-

This phase involves testing collection of modules which have been integrated into sub-system
Sub-system may be independently designed and implemented.
The most common problems which arise in large software systems are sub-system interface mismatches.
The sub-system test process should therefore concentrate on the detection of interface error by exercising these interfaces.

(4) System Testing:-

The sub-system is integrated to make the entire system.
The testing process is concerned with finding error which normally results from unanticipated interaction between sub-system and components.
It is also concerned with validating that the system meets functional and non functional requirements.

(5) Acceptance Testing:-

This is the final test in the testing process before the system is accepted for operational use.
It is sometimes called ***alpha testing***
It involves testing the system data supplied by system procedure than simulated data developed as part of the testing process

- ✓ When a system is to be tested as software product a testing process called ***Beta tested***
- ✓ It involves delivering a system to a number of potential customers who agree to use that system and provide the feedback.
- ✓ Repairing the program defect may introduce new defect so testing should be repeated after the new system is modified.
- ✓ This is sometimes called regression testing.

Difference between black box Testing and White box Testing

- **Black Box Testing**
  - Testing of completed unit of functional code is known as black-box testing or functional testing because tester treats the object as a black-box.
  - They concern themselves with verifying specified input against expected output and not worrying about the logic of what goes on in between.
  - User acceptance testing (UAT) and system testing are classic example of black-box testing.
- **White box Testing**
  - White-box testing or glass-box structure testing relies on analyzing the code itself and internal logic of the software.
  - White-box testing is often, but not always the purview of programmers.
  - It uses techniques which range from highly technical or technology specific testing through to things like code inspections.
  - Although white-box technique can be used at any stage in a software product`s life cycle they tend to be found in unit testing activities.

Difference between ERROR and BUG

- **ERROR:-**

  Is an undesirable deviation from requirement.
  Basically it is done by people.
  Error is the difference between the expected and Actual Results.

BUG:-

An error found in the development environment before the product is shipped to the customer.
It is logically devition.it will not show any error but it will affect the system.

Testing Levels

The following are testing level

1. **Unit Testing**
2. **Integration Testing**
3. **System Testing**
4. **System Integration Testing**
5. **Regression Testing**
6. **Black Box Testing**
7. **White Box Testing**
8. **Alpha Testing**
9. **Beta Testing**

(1) Unit testing:-

- Unit testing refers to test that verify the functionality of a specific section of code usually at the functional level. The object oriented environment this is usually at the class level and the minimal unit tests include the constructors and destructors.
- These type of tests are usually written by developer as they work on code(White-box style)
- One function might have multiple tests, to catch corner cases or other branches in the code.
- Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building block the software uses work independently of each other.

(2) Integration Testing

- Integration Testing is any type of software testing that seeks to verify the interfaces between components against a software design.
- Software component may be integrated in an iterative way or all together ("big bang").
- Normally the former is considered a better practice since it allows interface issues to be localized more quickly and fixed.
- Integration testing work to expose defect in the interface and interaction between integration components (module).
- Progressively larger group of tested software component corresponding to element of the architectural design are tested unit the software works as a system.

(3) System Testing

- System testing tests a completely integrated system to verify that it meets its requirements.

(4) System Integration Testing

- System Integration Testing verifies that a system is integrated to any external or third party systems defined in the system requirements.

(5) Regression Testing

- Regression testing focuses on finding defect after a major code changes has occurred.
- Specifically it seeks to uncover software regression or pld bugs that have come back such regression occur whenever software functionally that was previously working correctly stop working as intended.
- Typically regression occurs as an unintended consequence of program changes, when the newly developed part of the software collides with the previously existing code.
- Common method of regression testing include re-running previously run tests and checking whether previous fixed faults have re-emerged.
- The depth of testing depend on the in the release process and the risk of the added features.
- They can either be complete ,for changes.

(6) Black box Testing

- Black box testing treats the software as a “black box” without any knowledge of internal implementation. Black box testing method includes.
- This is otherwise called as functional testing.
- In contrary to white box testing here the person who is doing the black box testing need not have the programming knowledge.
- This is because the person doing the black box testing would access the output or outcomes as the end user would access and would perform though functionality testing to check whether the develop module or product behave in functionality in the way it has to be.

(6) White Box Testing

- Whit box testing focuses on designing test that view component as transparent.
- The implemetioanl detail of the components is known and that knowledge is used in test design and creation of test data.
- Other terms for white box testing are structural testing and clear box testing.

(7) Alpha Testing

- Alpha testing is simulated or actual operational testing by potential user/customer or an independent test team at the developer’s site.
- Alpha testing is often employed for off-the shelf software as a form of internal acceptance testing before the software goes to beat testing

(8) Beta Testing

- Beta testing comes after alpha testing.
- Versions of the software known as beta version are released to a limited audience outside of the programming team.
- The software is released to group of people so that further testing can ensure the product has few faults or bugs.
- Sometimes beta version is made available to the open public to increase the feedback field to a maximal number of future users.

Special System Testing

Types of Test

- Peak load test
- Storage testing
- Performance time testing
- Recovery testing
- Procedure testing
- Human factor testing
- **Peak load testing :-**
  - Determine whether the system will handle the volume of activities that occur when system is at the peak of its processing demand.
  - Example: all terminals are active at the same time.
- **Storage Testing:-**
  - Determine the capacity of the system to store transaction data on a disk or in other files.
  - Example:-verify documentation statement that the system will store 10,000 records of 383 byte length on a single flexible disk.
- **Performance time testing:-**
  - Determine the length of time used by the system to process transaction data.
  - Example:- response time for inquiry when system is fully loaded with operating data.
- **Recovery testing:-**
  - Determine ability of user to recover data restart system after failure,
  - Example:- load backup copy of data and resume processing without data or integrity loss.
- **Procedure Testing:-**
  - Determine clarity of documentation on operation and use system by having user do exactly what manuals request.
  - Example:-powering down system at end of week or responding to paper out light on paper.
- **Human factor testing:-**
  - Determine how user will use the system when processing data or preparing reports.
  - Example:-activities of user when there is not an immediate response to an inquiry.

Conversion

- Conversion is the process of changing from the old system to the new one.
- It must be carefully planned and executed.
- In general system conversion should be accomplished as quickly as possible.
- Long conversion periods increase tile possible frustration and difficulty of the task for all person involved including both analyst and users.

CONVERSION METHODS

These are four method of handling system conversion

1. **Parallel system**
2. **Direct conversion /cutover**
3. **Pilot system**
4. **Phase-in method**

✓ Each of method should be considered in light of the opportunities that it offer and problems that it may cause

(1) Parallel system

- The most secure method of converting an old to new system is to run both system in parallel.
- Hare users continue to operate the old new system as well the new system.
- This method is the safest conversion approach since it guarantees that should problems such as error in processing or inability to handle certain types of transaction arise in using the new system the organization can still fall back to the old system without loss of time or service.
- The other common approach is to create a case study example that include all frequently encountered situation that the system is able to handle and that the user should be able to handle then the user must use system to handle the actual situation that is the case study should require the user to pose and receive response to inquiries.
- During training system personal should be alert to comments made by user or to problem that users may encounter.
- Although human factor testing performed earlier are intended to detect difficulty some problem may not occur unit in experienced user are directly interacting with the system.

THE DISADVANTEGE:-

- The system cost double since there are two sets of systems cost .
- The fact that users know they can fall back to the old ways may be a disadvantage if there is potential resistance to the change or if users prefer the old system, in other words the new system may not get fair traial.

THE ADVANTEGES:-

- The parallel method offers greatest security.
- The old system can take if error are found in the new system.

(2) DIRECT COTOVER

- The direct cut over method convert from the old to the new system.
- The old system is used until a planned conversion day when it is replaced by the new system.
- The organization relies fully on the new system.
- If the analyst must make the change and wants to ensure that the new system fully replaces the old one that user do not rely on the previous method direct cutover will accomplish this goal.
- Ti forces all users to make the new system work they do not have any other method to fall back on.

The advantage:

- Force user to make new system work.
- There are immediate benefits from new method and controls.

The disadvantage

- There is no other system to fall back on if difficulties arise with new system.
- Require the most careful planning.
- Training session must be scheduled and maintained.
- The installation of all equipment must be on time.
- Direct conversion is quite common.

For example:

- A hotel operation decided to install an automated reservation system.
- The entire system was implemented during a one week period, when the computer system was set up s/w loaded and system tested.

- During that week a separate training worked with all the accounting personal to familiarize them with the operation and use of the system. These activities occurred Monday to Saturday
- On Sunday all personal were brought it to enter reservation guest charges and accounting information into new system so that it coincided with the current system.
- On Sunday evening after the close of business for the day the new system was started and uses permanently.
- The old paper reservation files were removed and the case register and book keeping machine were replace with the terminals.
- The new system became live at midnight on Sunday.
- There was no old system to fall back on.

(3) PILOT SYSTEM

- When the system involve new techniques or charge in organization perfumes, the pilot approach if often prefer.
- In this method a working version of the system is implemented in one part of the organization such as single work area or department.
- Base on feedback changes are made and the system is installed in the rest of the organization by one of the other methods.
- The user in the area typically knows n that they piloting a new system and the changes can he made to improve the system.
- When the system is deemed complete, it is installing throughout organization either all at once or gradually.
- The advantage: provided experience and live test before implementation.
- Disadvantage: may give the impression that the old system is unreliable and not error free.

(4) PHASE IN METHOD

- The phase in method is use when it is not possible to install a new system through an organization all at once.
- Implementation system across all users.
- The conversion of file, training of personal or arrival of equipment may force the staging of the implementation over a period of time, ranging week to months.

THE ADVANTAEGES

- Allow some user to take advantages of the system early.
- Allow training and installation unnecessary resources.

For example

- A medical system aimed at linking 10 or 15 different clinics to a hospital may phase it over a year.
- The work required to cover patient and insurance record on paper to file store on magnetic disks.
- Require 2 to 3 weeks for each clinic.
- Therefore the analyst may phase this system in one clinic at a time allowing 3 to 4 weeks for conversion.
- It is conceivable in this system that the full conversion will be phase over one year.