UNIT-III

# Estimation & Scheduling

## ★ Software Scope -

→ • Software scope describes
  i) Functions & features that are to be delivered to end users
  ii) Data i.e input & output
  iii) Content that is presented to users
  iv) Performance, constraints, interfaces & reliability

• In scope description, various functions are described.

• These functions are evaluated and refined to provide more details before estimation

• For performance consideration, processing and response time requirements are analyzed

• After identifying scope, its feasibility must be ensured

• There are 4 dimensions of software feasibility:
  1) Technology
  2) Finance
  3) Time
  4) Resource.

Date
Page

## * Software Project Estimation -

→ • Software estimation helps in understanding scope of project.

• Accurate estimation of project size, duration, cost is important, because it helps in quoting appropriate project cost to customer.

• Steps for software estimation -

i) Estimate the size of development product. This generally ends in LOC or FP.

ii) Estimate the effort in person-months or person-hrs

iii) Estimate the duration in calendar months.

iv) Estimate the project cost in dollars / local currency.

## ★ Function Point (FP) Model :-

→ • FP model is based on functionality of delivered application

• They are independent of programming language

• FP are derived using:
1) Countable measure of soft. req. domain
2) Assessments of software complexity.

• Data for following information domain are collected :
1) No. of user inputs
2) No. of user outputs
3) No. of user inquiries
4) No. of files
5) No. of external interfaces

## ★ LOC based Estimation –

→ • Size oriented measure is derived by considering size of software that is produced

• It is direct measure of software

• Size measure is based on lines of code computation

• The line of code is defined as one line of text in source file.

• Simple set of size measure is :
1) Size = Kilo Lines of code (KLOC)
2) Effort = Person/month
3) Productivity = KLOC/person-month
4) Quality = No. of faults/KLOC
5) Cost = $/KLOC
6) Documentation = Pages of documentation/KLOC

• While counting LOC :
1) Don't count blank lines & comments

• Size oriented measure is not universally accepted method.

- Definition -

Project Decomposition :

It is the process of dividing components of large project into smaller portions called delivexables.

★ Cocomo - II Model -

→ • Applied for modern software development practices addressed for projects

• Sub - models :

1) Application Composition Model -

• Used during early stages of soft. engg., when prototyping of user interfaces, consideration of software, etc are paramount

2) Early Design Stage Model -

• Used once requirements have been stabilized and basic software architecture has been established

3) Reuse model -

• It considers systems that have significant amount of code which is reused from earlier software system

4) Post-architecture-stage model -
- Used during construction of software.

- Cocomo-II model uses object points

- Object point is indirect software measure that is computed using counts of no. of
  1) Screens
  2) Reports
  3) Components

★ Project Scheduling -

→ • Software project scheduling can be defined as activity that distributes estimated effort across planned project duration by allocating effort to specific soft.engg. tasks.

- Principles:

1) Compartmentalization -
Project must be compartmentalized into no. of manageable activities, tasks.

2) Interdependency -
Interdependency of each compartmentalized task must be determined.

3) Time Allocation –
Each task to be scheduled must be allocated some number of work units

4) Effort Validation •

5) Defined Responsibilities –
Every task should be assigned to specific team member

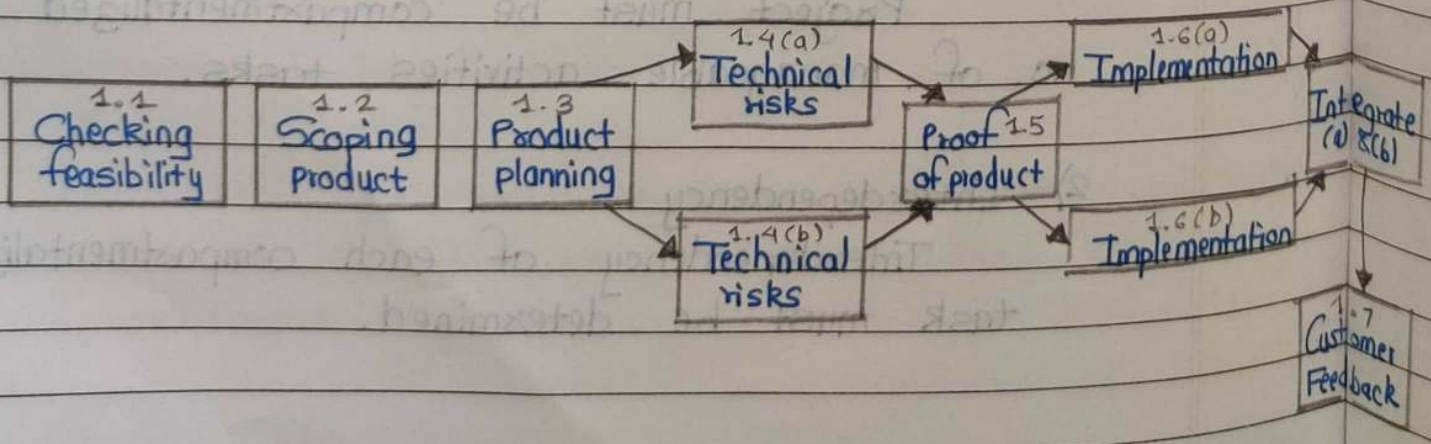
6) Defined Outcomes •

7) Defined Milestones

★ Task Network –

→ • Task is small unit of work

• Task network is graphical representation with:
1) Nodes corresponding to activities
2) Tasks are linked if they are dependent
3) Eg, task network for product development-



- Task network definition helps manager to understand project work breakdown structure

-

★ Time Line Chart -

→ • In software project scheduling the timeline chart is created

- Purpose of timeline chart is to emphasize scope of individual task.

- Hence, set of tasks are given as input to time line chart

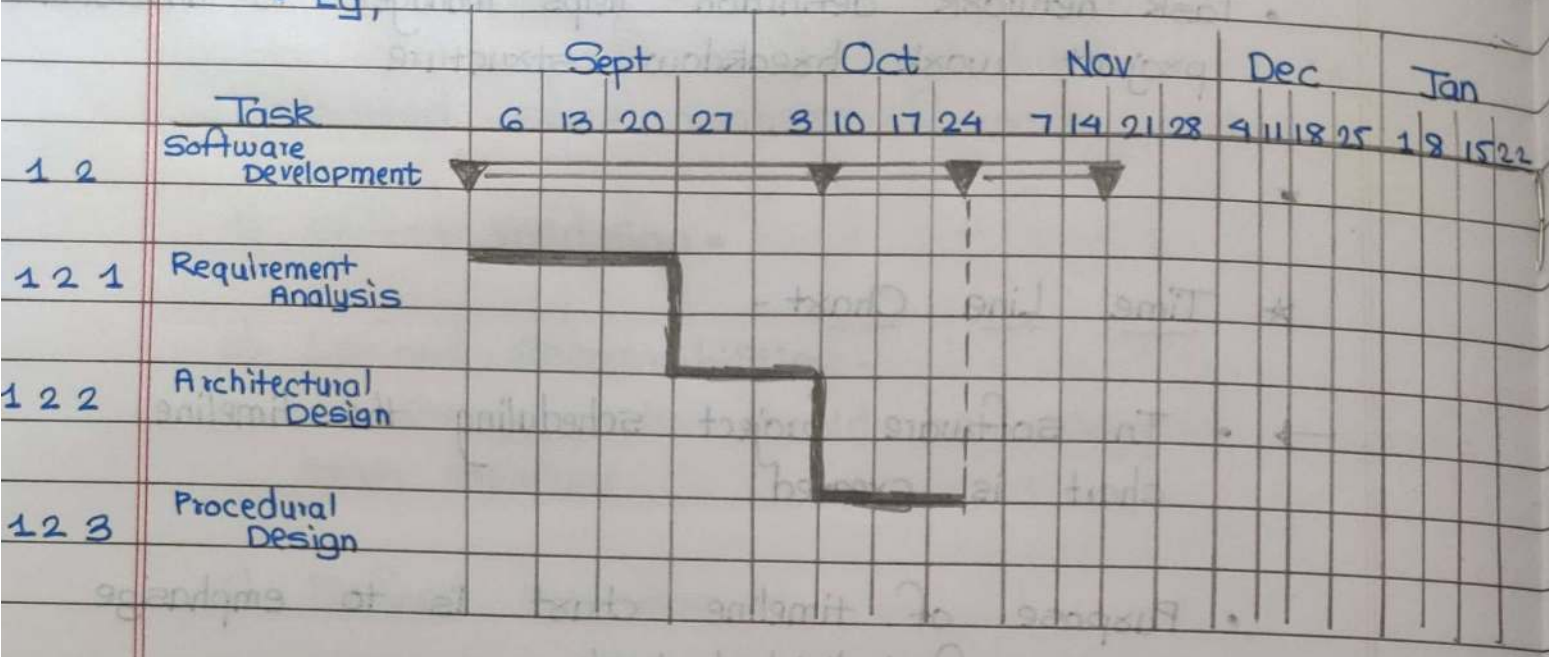
- In time line chart,
  1) All tasks are listed at leftmost column

  2) Horizontal bars indicate time required by corresponding task

  3) When multiple horizontal bars occur at same time, that means concurrency can be applied

  4) Diamonds indicate milestones

• Eg,


Sept
Oct
Nov
Dec
Jan
Task
6 13 20 27 3 10 17 24 7 14 21 28 4 11 18 25 1 8 15 22
1 2 Software Development
1 2 1 Requirement Analysis
1 2 2 Architectural Design
1 2 3 Procedural Design

# DESIGN ENGINEERING

- **Definition -**

  **Software Design -**

  It is model of software which translates requirements into finished software product in which details that are necessary to implement system are given.

- **Characteristics of well formed design -**
  1) Good design should implement all requirements specified by customer
  2) Design should be simple
  3) Design should be comprehensive

## ★ Design Quality Attributes -

→ • Design Quality Attributes popularly known as FURPS (Functionality, Usability, Reliability, Performance & Supportability) is set of criteria

| Quality Attribute | Meaning |
|---|---|
| 1) Functionality | Functionality can be checked by assessing set of features & capabilities of functions. |

| | |
|---|---|
| 2) Usability | Usability can be assessed by knowing usefulness of system |
| 3) Reliability | Reliability is measure of frequency and severity of failure |
| 4) Performance | It is measure that represents response of system |
| 5) Supportability | It is ability to adopt changes made in software |

## ★ Design Concepts ★

- Sub-topics -

★ Refinement -

→ • Refinement is actually a process of elaboration

- Architecture of program is developed by successively refining levels of procedural detail

- Abstraction & Refinement are complementary concepts.

## ★ Modularity -

→ • Software is divided into seperately named and addressable components called as Modules

• The effort (cost) to develop software module decreases as the no. of modules increases

• Meyer defined criteria to evaluate design method w.r.t its ability to define effective modular system
1) Modular Decomposition
2) Modular Composability
3) Modular Understandibility
4) Modular Continuity
5) Modular Protection

## ★ Architecture -

→ • Architecture means representation of overall structure of integrated system

• In this, various components interact and data of structure is used by various components

• In architectural design, various system models like Structural model, Dynamic model, Process model, etc are used.

## ★ Refactoring -

→ • Refactoring is necessary for simplifying the design without changing function or behaviour

• It is disciplined way to clean up code that minimizes chances of introducing bugs

• Benefits of refactoring are:
1) Redundancy is achieved
2) Inefficient algorithms can be eliminated / replaced
3) Inaccurate data structures can be removed
4) Other design failures can be rectified.

• Decision of refactoring particular component is taken by designer of software system.

## ★ Cohesion -

→ • With help of cohesion, Information hiding can be done

• Cohesive module performs only "one task" in software procedure with little interaction with other modules.

- Different types of cohesion are:
  1) Coincidentally cohesive
  2) Logically cohesive
  3) Temporal cohesion
  4) Procedural cohesion
  5) Communicational cohesion

- Goal is to achieve high cohesion for modules in system.

★ Coupling –

→ • Coupling effectively represents how modules can be "connected" with other module or outside world

- It is measure of interconnection among modules in program structure

- It depends on interface complexity between modules

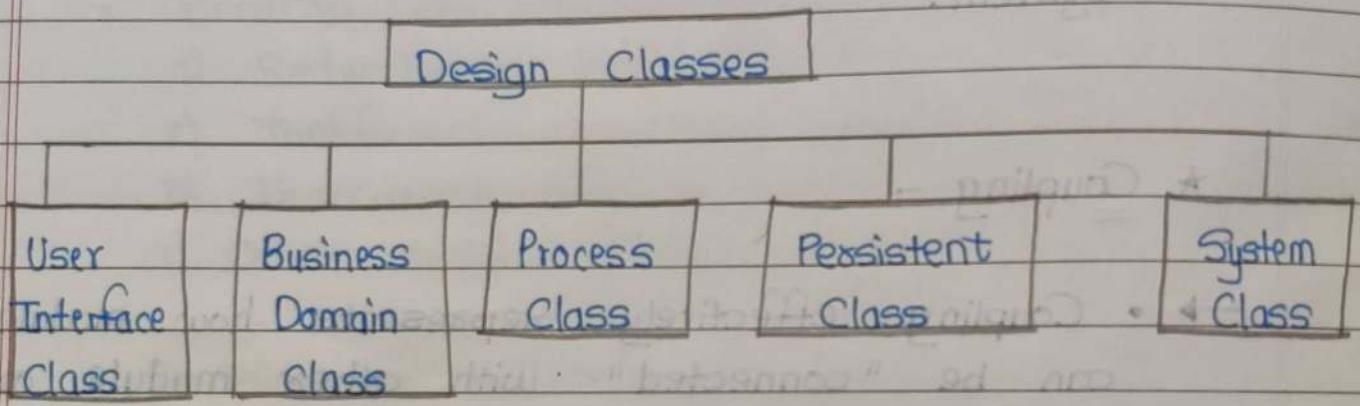
- Different types of coupling are:
  1) Data coupling
  2) Control coupling
  3) Common coupling
  4) Content coupling

- Goal is to achieve lowest possible coupling among modules

## ★ Design Classes -

→ • Design classes are described as classes that describe some elements of problem domain, focus on various aspects of problem from user's view

• Types of Design Classes -




1) User Interface Class -
It defines all abstractions necessary for Human - Computer Interface (HCI)

2) Business Domain Class -
These classes identify attributes and services that are needed to implement elements of business domain

3) Process Class -
Implements lower level abstraction used by business domain

4) Persistent Class -
Represents databases which will be retained as it is, after execution of software

5) System Class -
These are responsible for software management & control functions used for system operation.

★ Design Model -

→ • Design model can be viewed in two dimensions

• The process dimension indicates evolution of design model as design tasks are executed as part of software process

• The Abstraction dimension represents level of details as each element is transformed into design equivalent

■ Elements of Design Model -

(1) Data Design Elements :
• Data design represents high level of abstraction

• The data represented at data design level is refined gradually for implementing computer based system.

(2) Architectural Design Elements -

- Architectural design gives layout for overall view of software

(3) Interface Design Elements -

- Interface Design represents detailed design of software system.
- In this, how information flows from one component to other component of system is depicted.

(4) Component Level Design Elements -

- It describes the internal details of component.
- In this design, all local data object, required data structures & algorithmic details are exposed.

(5) Deployment Level Design Elements -

- It indicates how software functions and software subsystems are assigned to physical environment of product

★ Guidelines for Component Level Design -

→ 1) Components :
- Components are part of architectural model
- Component names should be specified from problem domain

2) Interfaces :
- Interface serve important role in communication & collaboration
- Interface should be drawn as circle with solid line connecting it to another element.
- It should flow from left side of component

3) Dependencies and ~~interfaces~~ : (Inheritance maybe)
- Dependencies must be shown from left to right
- Inheritance should be shown from bottom to top.

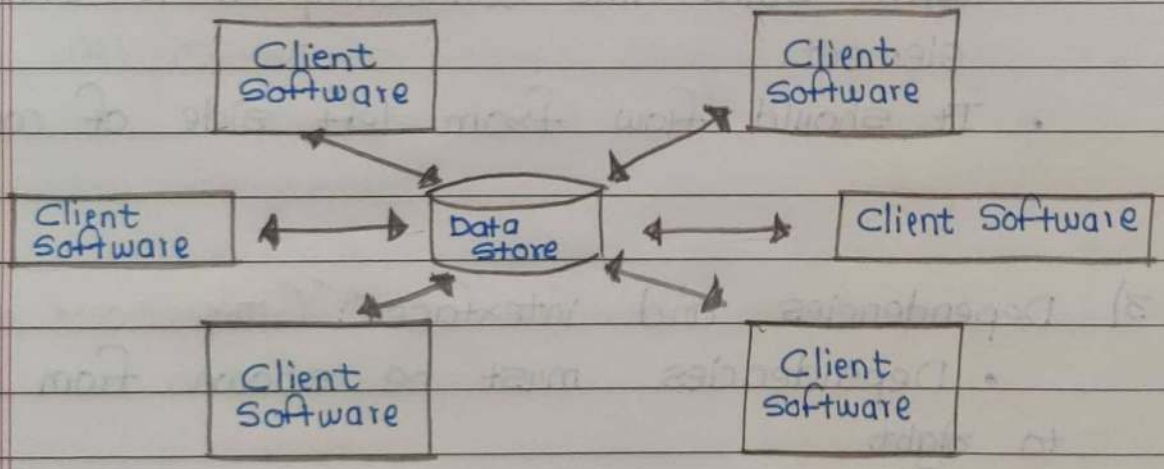
■ Definition -
Software Architecture -
It is structure of systems which consists of various components, externally visible properties of these components and inter-relationship among these components.

## * Data - Centered Architecture -

→ • In this architecture, data store lies in centre and other components frequently access it by performing add, delete & modify operations

• Client software requests for data to the central repository.

• Diagram -




• Data centered architecture possess property of interchangeability.

• In data centered architecture,
Components are : Tables, queries
Communication is : by relationship

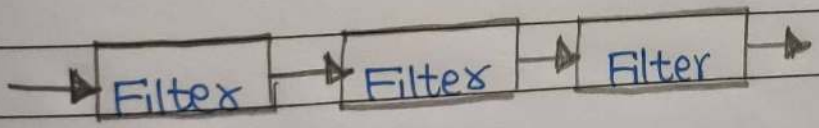
## ★ Data-flow Architecture Style -

→ • In this, series of transformations are applied to produce data

• Set of components called filters are connected by pipes to transform data from one component to other.

• Diagram -


• If data flow degenerates into single line of transforms, it is termed as batch sequential



UNIT - V

# Risk & Configuration Management

- **Definition -**

  **Risk management -**

  It refers to process of making decisions based on evaluation of factors that threats to business.

  Types of risks:

  1) Project risk
  2) Technical risk
  3) Business risk

★ **Risk Identification -**

→ • It is defined as efforts taken to specify threats to project plan

• Risk Identification can be done by identifying known and predictible risks

• Risk Identification is based on 2 approaches,

1) Generic Risk Identification -

   It includes potential threat identification to project

2) Product - specific risk identification -

   It is done by understanding people, technology & environment in which product is built.

## ★ Risk Assessment Process -

→ • The best approach is to prepare set of questions that can be answered by project manager in order to assess project risks

• These questions can be,
1) Will project get proper support by manager?
2) Is there clear understanding of requirements?
3) Expectations set for product are realistic?
4) Is project scope stable?
5) Are project requirements stable?

• Thus, no. of negative answers represents severity of impact of risk on overall project.

## ★ Risk Projection using Risk Table -

→ • Building risk table is simplest & most commonly used technique to project risks

• Eg,

Risk Table

| Risk | Category | Probability | Impact | Rmmm |
|---|---|---|---|---|
| Is skilled staff available | Staff | 50% | Catastrophic | |
| Is team size sufficient | Staff | 62% | Critical | |

| | | | | |
|---|---|---|---|---|
| Have staff received sufficient training | Staff | 25% | Marginal | |
| Is software management tool available | Environment | 30% | Negligible | |

- While building risk table,
  1) Project team first enlists all probable risks, Each risk is then categorized
  2) Probability of occurrence of each risk is estimated by team member
  3) Impact of each risk is assessed
- After building table, it is then sorted by probability & impact
  This arrangement is first-order prioritization
- Project manager goes through this table and draws cut-off line at some point in table.
- Risk above the cut-off line should be managed.

## ★ Risk Mitigation, Monitoring & Management (RMMM) -

→ • RMMM stands for risk mitigation, monitoring & management.

1) Risk Mitigation -
- Risk mitigation means preventing risks to occur
- Following are some steps to be taken for mitigating risks
  i) Communicate with staff to find risk
  ii) Find out & eliminate all causes that can create risk
  iii) Conduct timely reviews to speed up work
  iv) Provide additional staff for conducting critical activity.

2) Risk Monitoring -
- In risk monitoring, following things must be monitored by project manager,
  i) Approach or behaviour of team members
  ii) Type of co-operation among team members
  iii) Type of problems that are occurring, etc.

- Objective of risk monitoring is,
  i) To check whether predicted risks really occur or not
  ii) To ensure steps defined to avoid risk are applied properly or not
  iii) To gather information for analyzing the risk.

3) Risk Management -

- Project manager performs this task when risk becomes a reality
- If manager is successfully in applying project mitigation effictively then it becomes very much easy to manage risk.

★ Rmmm Plan -

→ • Rmmm plan is document in which all risk analysis activities are described

- Project manager includes this document as part of overall project plan
- Typical template for Rmmm plan -

| Risk Information Sheet | | | |
|---|---|---|---|
| Project name - | | | |
| Risk id | Date | Probability | Impact |
| | | | |
| Origin | | Assigned to | |
| Description - | | | |
| Refinement / Context - | | | |
| Mitigation / Monitoring - | | | |
| Contingency plan - | | | |
| Status - | | | |
| Approval | | Closing Date | |

## ★ Software Configuration Management (SCM) -

→ • Software Configuration Management is set of activities carried out for identifying, organizing and controlling changes throughout lifecycle of computer software.

- During development of software, change must be managed and controlled in order to improve quality & reduce error
- Hence SCM is quality assurance activity that is applied throughout process
- The SCM is concerned with manging evolving software systems
- SCM is set of tracking and control activities that begin when software development project begins and terminates when project terminates

## ★ Scm Repository -

→ • Software repository is collection of information accessed by software engineers to make appropriate changes in it.

• This repository is handled using all modern database management functions

• It must maintain data integrity, sharing & integration

• Features are -

1) Versioning -
Repository must be able to maintain all versions of individual work products

2) Dependency tracking & change management -
Repository must have ability to maintain integrity

3) Requirements Tracing -
Repository must have ability to trace requirement from constructed components

4) Configuration Management -
Repository must keep track of series of configurations

5) Audit trails -
It establishes additional information about changes made

## * Layers of scm Process -

→ 1) Configuration Identification -

- Software configuration items must be seperately named and identified as object.
- These objects must be arranged using object oriented approach.

- There are 2 categories of object - Basic objects & Aggregate objects

2) Change Control -

- Changes in any software projects are vital
- Sometimes, introducing small changes in system may lead to big problems

- For managing such changes, human procedures or automated tools can be used.

3) Version Control -

- Version is instance of system which is functionally distinct in some way from other instances

- Version control works to help manage different versions of configuration items during development process.

4) Configuration Audit -

- In order to ensure that change has been properly implemented or not, two activities are carried out
  1) Formal Technical Review (FTR)
  2) Software Configuration Audit

5) Status Reporting -

It focuses on communication of changes to all people in organization that involve with changes.

UNIT - VI

# SOFTWARE TESTING

## ★ Software Testing -

→ • Software testing is critical element of software quality assurance & represents ultimate review of specification, design & coding

• Testing is set of activities that can be planned in advance & conducted systematically

• If testing is conducted haphazardly, it may lead to many undetected errors in system being developed

• Hence performing testing by strategies is very much essential in development of software

• Principles -

1) All tests should be traceable to customer requirements
2) Tests should be planned long before testing begins
3) Exhaustive testing is not possible
4) Testing should be conducted by independent third party.

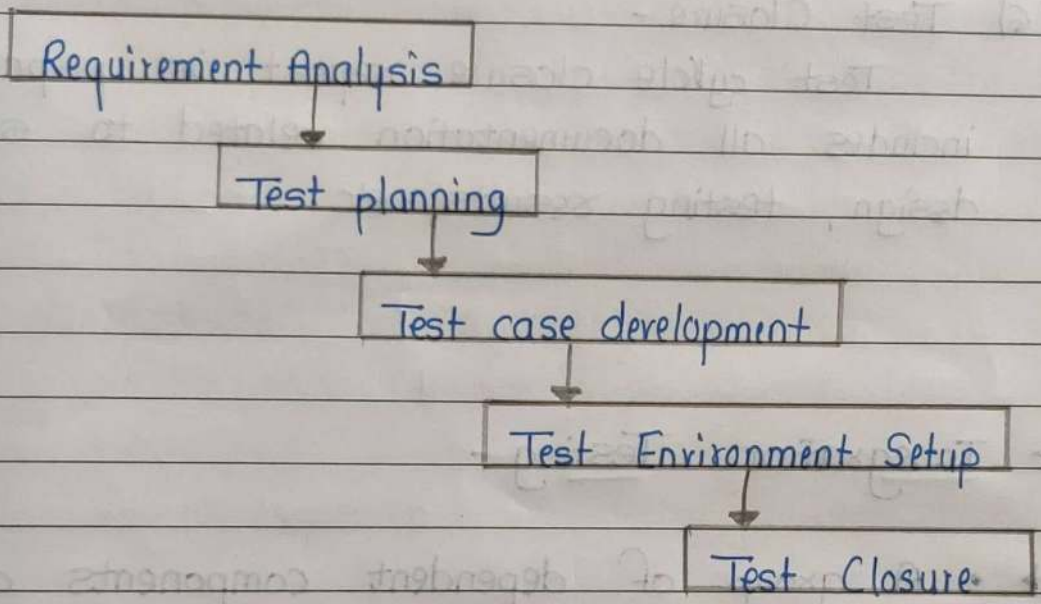
★ Difference - Verification v/s Validation

| Verification | Validation |
|---|---|
| i) It refers to set of activities that ensure software correctly implements a function | i) Validation refers to set of activities that ensure that software has been built is traceable to customer requirements |
| ii) After valid specification, verification starts | ii) Validation begins as soon as project starts |
| iii) It is for prevention of errors | iii) It is for detection of errors |
| iv) It is conducted using reviews, walkthroughs, audits, etc | iv) It is conducted using system testing, stress testing, etc |
| v) It is also termed as white box / static testing | v) It is also termed as black box / dynamic testing |
| vi) It finds out about 50-60% of defects | vi) It finds out about 20-30% of defects |
| vii) It is about process, standard & guideline | vii) It is about product |

## ★ Software Testing Life Cycle (STLC) -

→ • STLC is sequence of activities conducted during testing process.

• Phases in STLC model -




1) Requirement Analysis -
In this phase, it is identified that requirements are testable or not

2) Test planning -
Manager of testing team calculates estimated effort and cost for testing work

3) Test Case Development -
Test cases are created & test data is identified

4) Test Environment Setup -
Decides hardware and software setup under

which work product is tested

5) Test Execution -
Testing team starts case development and execution activity

6) Test Closure -
Test cycle closure repost is prepared which includes all documentation related to software design, testing results, etc.

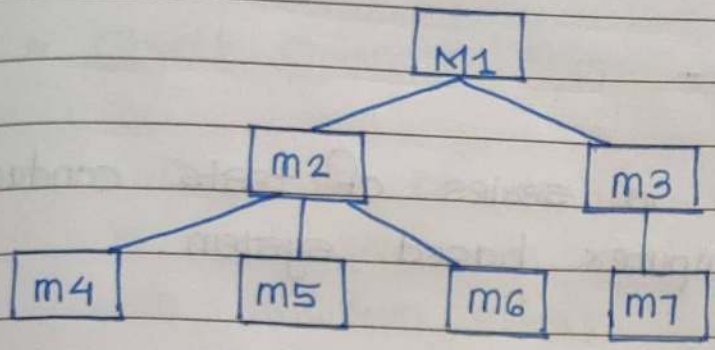
* Integration Testing -

→ • A group of dependent components are tested together to ensure their quality of integration unit.

• Objective is to take unit tested components and build a program that is dictated by software design.

• Approaches in Integration Testing:
1) Top - Down Integration -
• It is incremental approach in which modules are integrated by moving down control structure

• Eg,

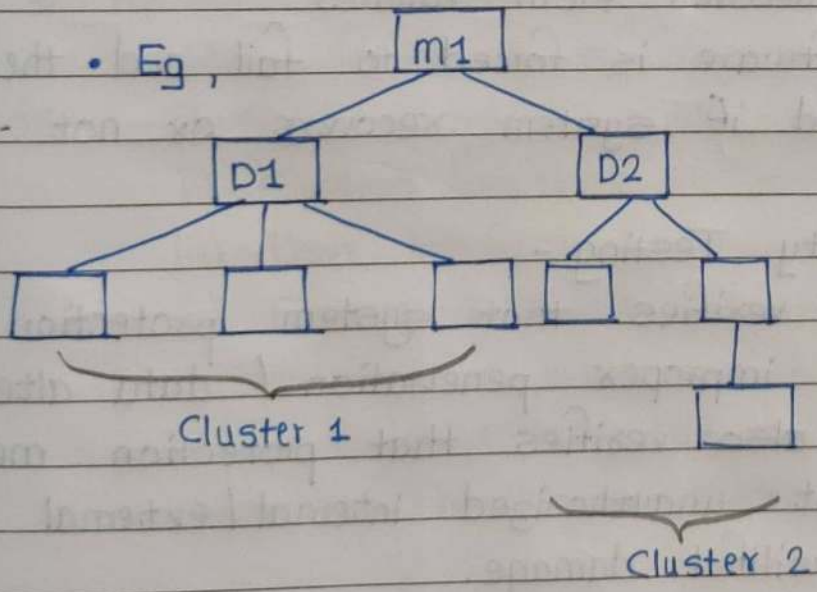



In top-down approach, if DFS approach is adopted then sequence of module integration will be m1 - m2 - m4 - m5 - m6 - m3 - m7

2) Bottom-Up Integration -

- In this, modules at lowest levels are integrated first, then integration is done by moving upward.

- Eg,




Components are collected together to form cluster 1 & 2. Then each cluster is tested using driver program.

## ★ System Testing -

→ • System Test is series of tests conducted to fully computer based system.

• Main focus of this is to test,
1) System functions
2) System reliability
3) System installation
4) System behaviour (Stress test)
5) Hardware & Software Integration

• Types of system test are:

1) Recovery Testing -
• It is intended to check system's ability to recover from failures
• Software is forced to fail and then it is verified if system recovers or not

2) Security Testing -
• It verifies that system protection mechanism prevent improper penetration / data alteration
• It also verifies that protection mechanism prevent unauthorized internal / external access or willful damage.

## ★ Object Oriented Testing -

→ • Objective of testing for object oriented software is to uncover as much errors as possible with minimum efforts in realistic time span

• The strategy is to start testing in small and work outward to testing in large

• Basic unit of testing is class that contains attributes & operations

## ★ Validation Testing Types -

→ 1) Validation Testing -

• Main focus is to uncover errors in
  i) System input/output
  ii) System functions
  iii) User interfaces
  iv) System behaviour

• Software validation can be performed through series of black box tests

2) Acceptance Testing -

• It is kind of testing conducted to ensure software works correctly in user work environment

• Types are :

i) Alpha Test -
Version of complete software is tested by customer under supervision of developer

ii) Beta Test -
Version of complete software is tested by customer without presence of developer.

3) Smoke Testing -
- Integration testing technique, used for time critical projects wherein project needs to be assessed frequently.

★ Difference - Alpha Testing v/s Beta Testing

| → Alpha Testing | Beta Testing |
|---|---|
| i) Performed at developer's site | i) Performed at end user's site |
| ii) Performed in controlled environment as developer is present | ii) Performed in uncontrolled environment as developer is absent |
| iii) Less probability of finding errors | iii) High probability of finding errors |
| iv) Not considered as live application | iv) Considered as live application |