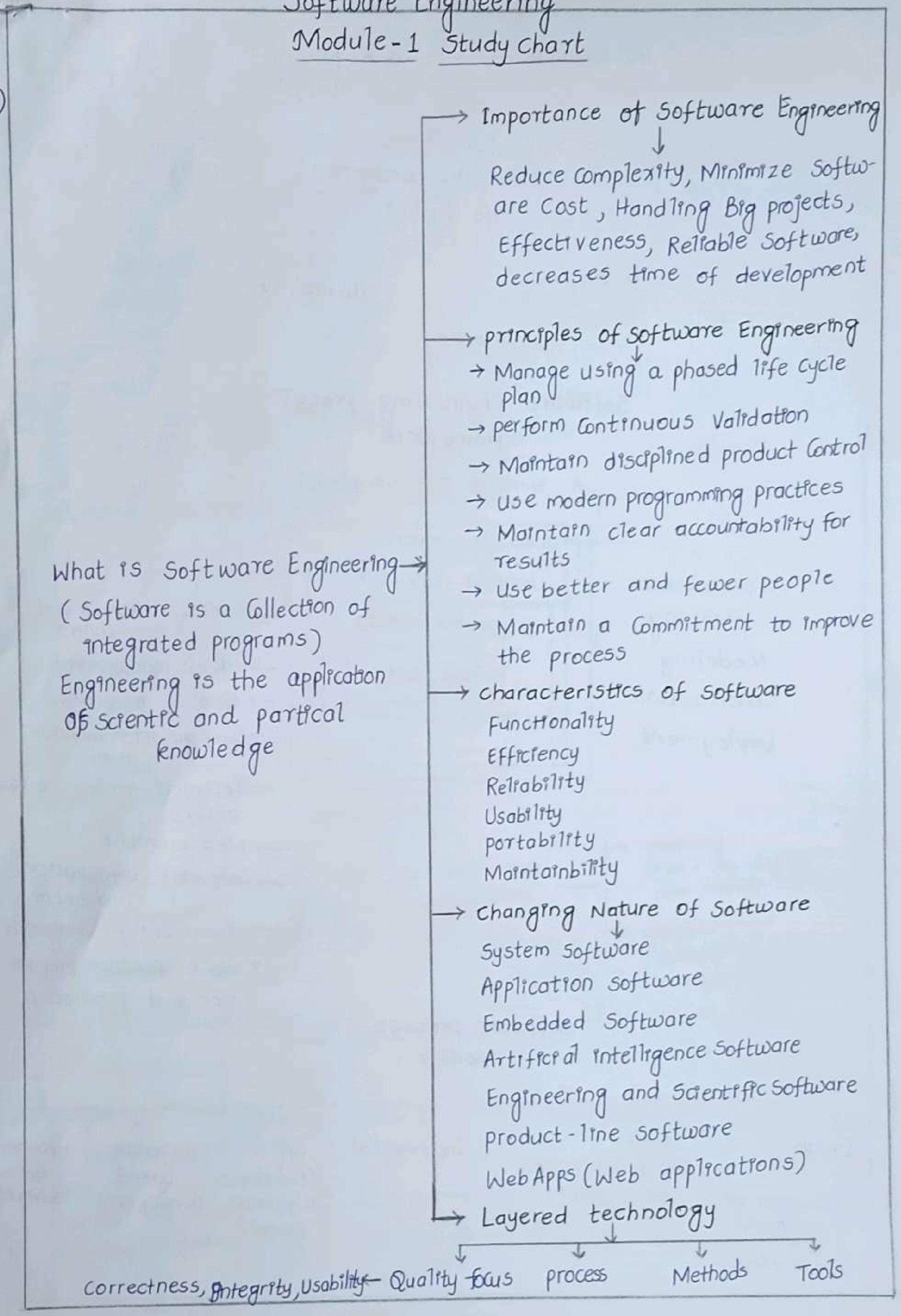
Software Engineering
Module-1 Study Chart

1)



What is Software Engineering
(Software is a Collection of integrated programs)
Engineering is the application of scientic and partical knowledge

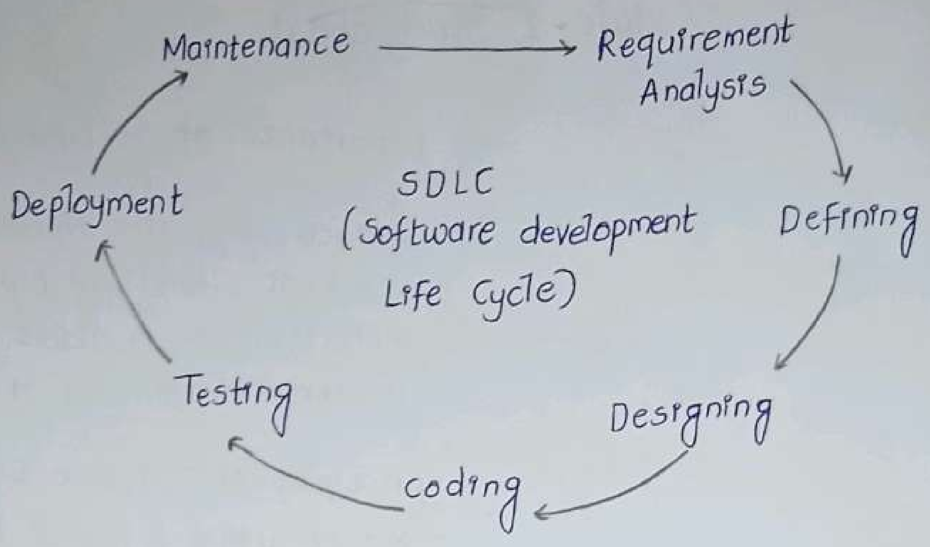
→ Importance of Software Engineering
↓
Reduce Complexity, Minimize Software Cost, Handling Big projects, Effectiveness, Reliable Software, decreases time of development

→ principles of software Engineering
- → Manage using a phased life cycle plan
- → perform Continuous Validation
- → Maintain disciplined product Control
- → use modern programming practices
- → Maintain clear accountability for results
- → Use better and fewer people
- → Maintain a Commitment to improve the process

→ characteristics of software
- Functionality
- Efficiency
- Reliability
- Usability
- portability
- Maintainbility

→ Changing Nature of Software
↓
- System Software
- Application software
- Embedded Software
- Artificial intelligence Software
- Engineering and Scientific Software
- product-line software
- WebApps (Web applications)

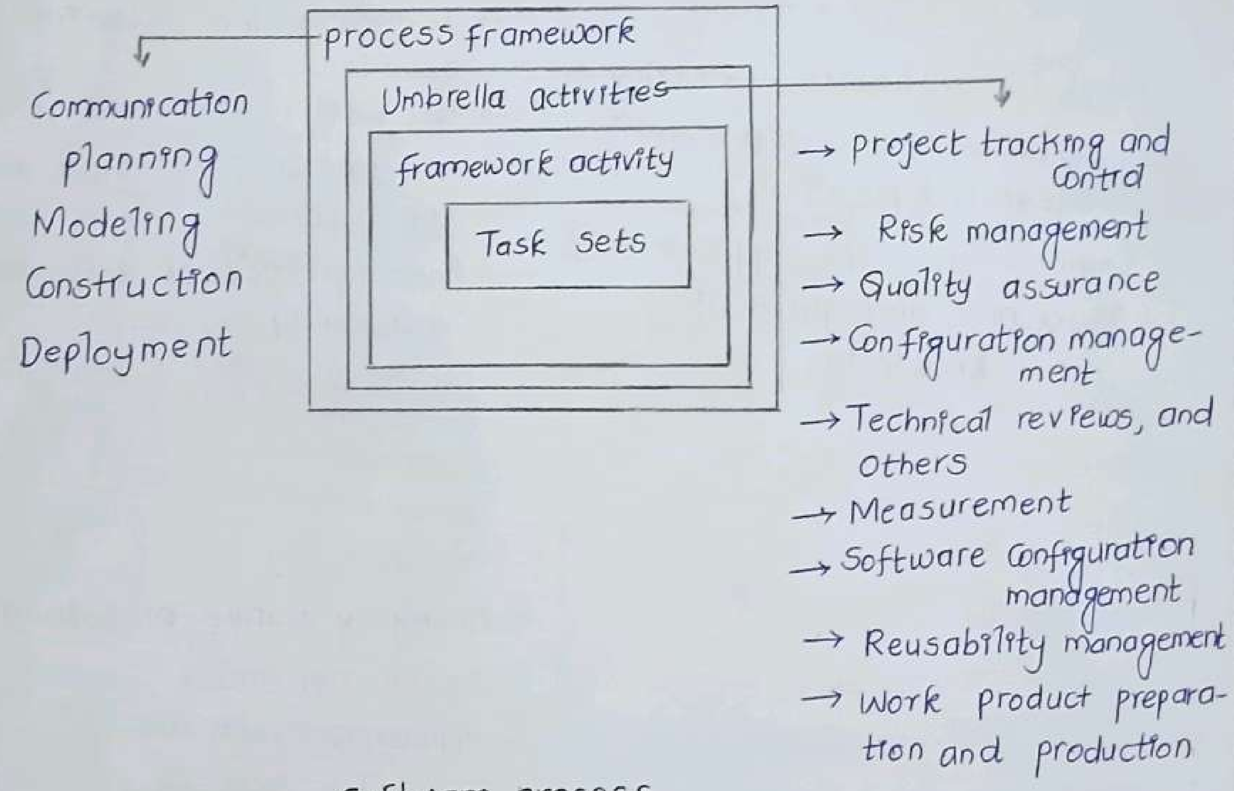
→ Layered technology
↓
Quality focus — Correctness, Integrity, Usability
process
Methods
Tools

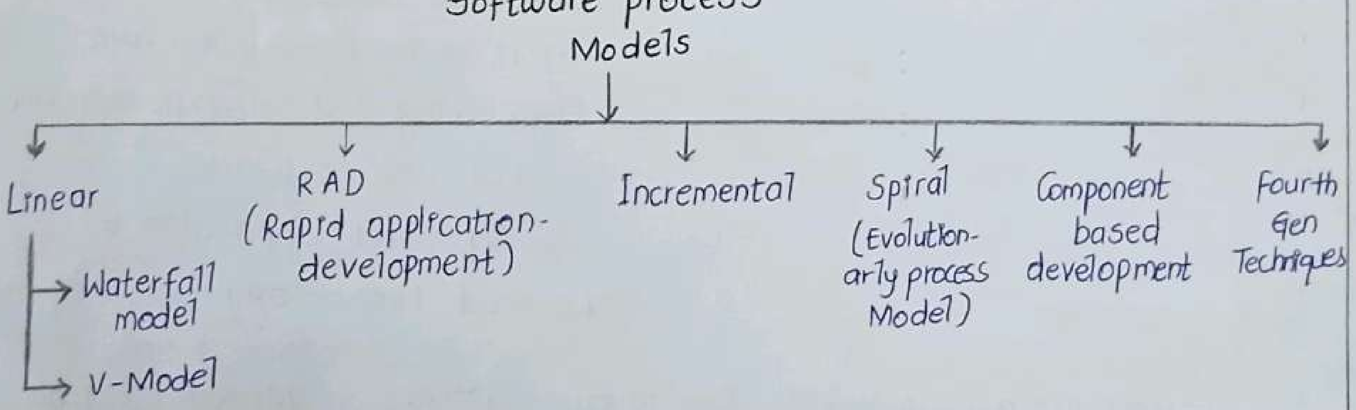


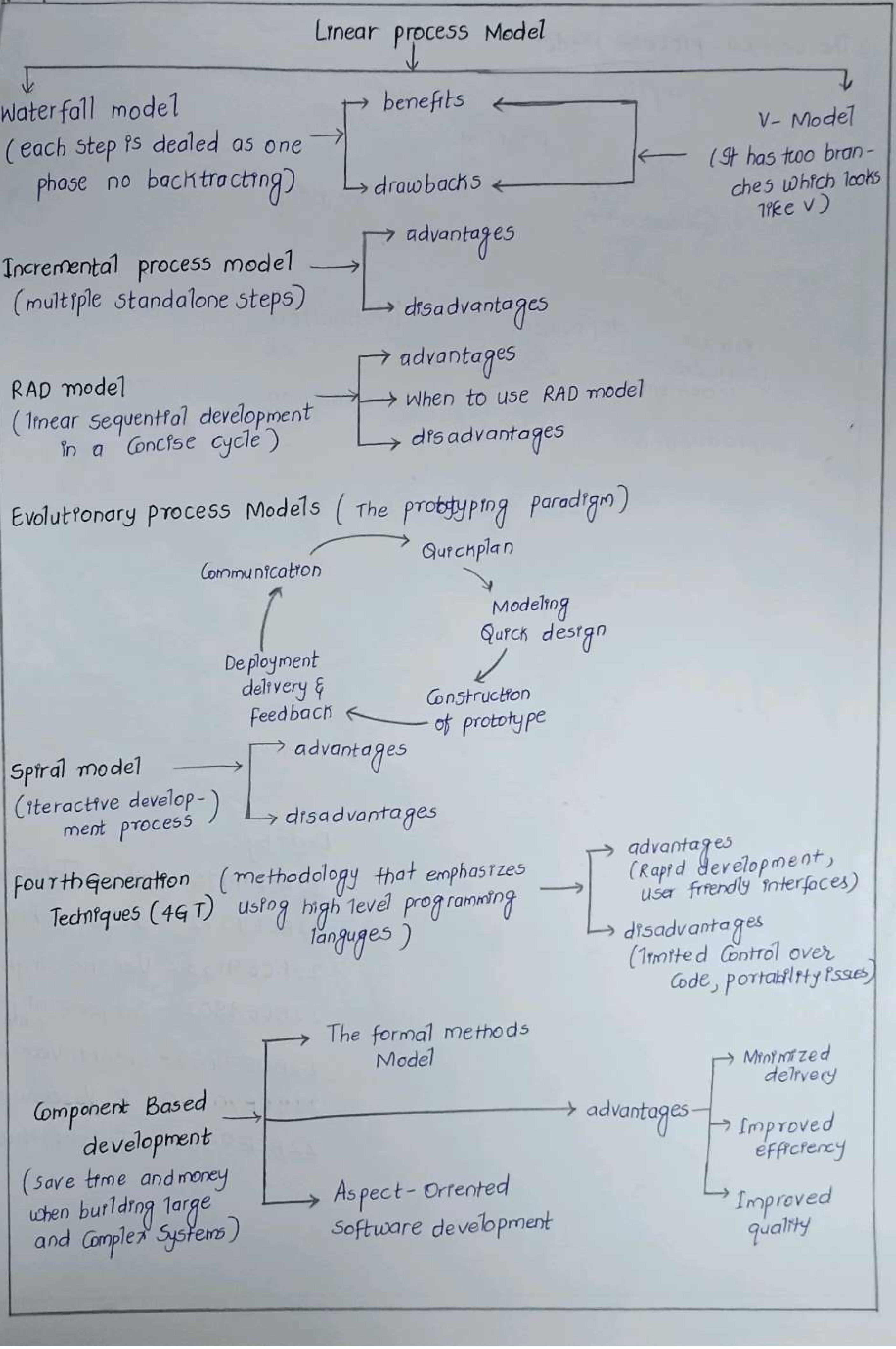
2)

Software Engineering process Frame Work

A Generic process Model




Software process Models
Linear
→ Waterfall model
→ V-Model
RAD (Rapid application-development)
Incremental
Spiral (Evolution-arly process Model)
Component based development
Fourth Gen Techniques



Linear process Model

Waterfall model (each step is dealed as one phase no backtracting)

V- Model (It has too branches which looks like V)

benefits

drawbacks

Incremental process model (multiple standalone steps)

advantages

disadvantages

RAD model (linear sequential development in a Concise cycle)

advantages

When to use RAD model

disadvantages

Evolutionary process Models (The prototyping paradigm)

Communication

Quickplan

Modeling Quick design

Construction of prototype

Deployment delivery & feedback

Spiral model (iteractive development process)

advantages

disadvantages

Fourth Generation Techniques (4GT) (methodology that emphasizes using high level programming languges)

advantages (Rapid development, user friendly interfaces)

disadvantages (limited Control over Code, portability issues)

Component Based development (save time and money when building large and Complex Systems)

The formal methods Model

advantages

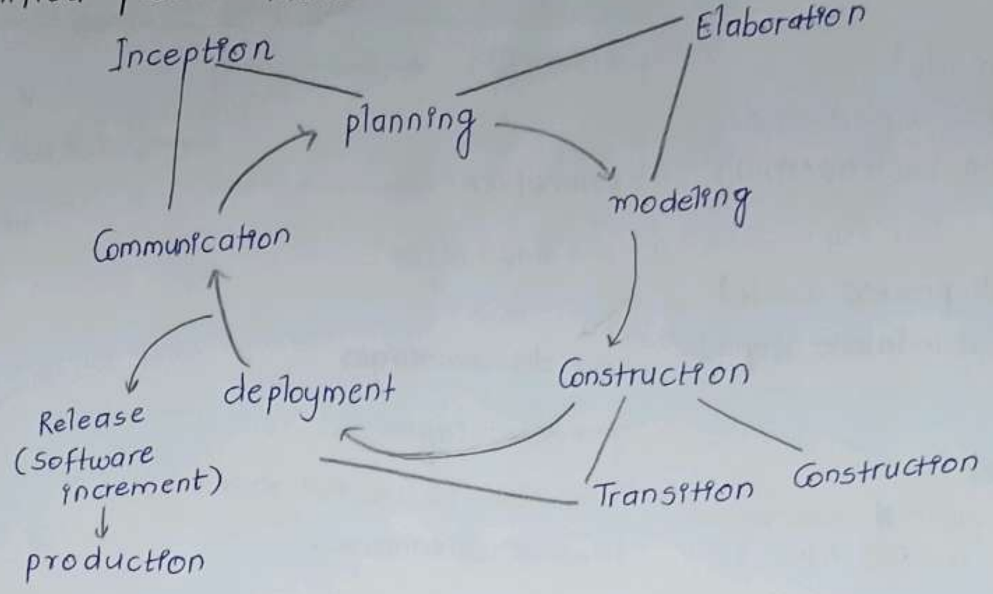
Minimized delivery

Improved efficiency

Improved quality

Aspect-Oriented Software development

The Unified process Model



Done by :-

22BCE9976	Kalluru Jahnavi
22BCE9572	J.Suvarchala
22BCE9133	Varsha Singh
22BCE8802	Sanjeevani pandit
22BCE7455	Srishti Verma
22BCE20390	P. Yasaswini
22BCE9587	T. Likhitha

Module-2 Study Chart

Requirements Engineering

(This is the process of identifying, eliciting, analyzing, specyfing, Validating and managing the needs and expectations of stakeholders for a Software System)

- Feasibility study (or) Inception
  (The process begins by examining the problem)
- Elicitation
  (Specifying a basic set of requirements)
- Elaboration
  - → Scenario-based elements
  - → class-based elements
  - → Behavioral elements
  - → flow-oriented elements
- Requirements Verification and Validation
- Requirements Management

What is UML?

(Unified Modeling Language)

(It is general purpose, graphical modeling language in the field of Software Engineering)

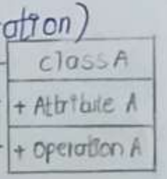
- Things
  - → Structural
    - [class, interface, active class, use case, component]
  - → Behavioral
    - [Interaction, state machine]
  - → Grouping
    - package
  - → Annotational
    - Note
- Relationships
  - Dependency
  - Generalization
  - Associations
  - Realization
- Diagrams
  1) class
  2) object
  3) Sequence
  4) collaboration
  5) Usecase
  6) statechart
  7) Activity
  8) Component
  9) Deployment

Relationships

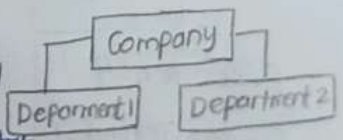
- Generalization (It Connects Specialized element with Generalized element)
  - Inheritance/Generalization
  - ————▷
- Realization (It is a relationship in which two elements are Connected)
  - Realization
  - - - - - - - - ->
- Association (It is a set of links that Connects elements)
  - Association
    ————
  - Direct Association
    ————>
  - Aggregation
    ————◇
- Dependency (this is a relationship between two things in which one can affect other)
  - Dependency
  - - - - - - - - ▷

Types of UML Diagrams

- Structural diagrams (static aspect of the System)
  - class diagram (Static view of an application)
    - class Name
    - class Attributes
    - class operations



  - Object diagram (defines external behavior without revealing internal structure of System) [ellipse - user, System boundary rectangle, stick person-user]



  - Component diagram (Displays the structural relationship of Components, used in complex Systems)
  - Deployment diagram (It shows the hardware of your system and its Software in that hardware)
- Behavioral diagrams (captures dynamic aspect of System)
  - Usecase diagram (This gives a graphic overview of the actors involved in a system)
  - Sequence diagram (shows how objects interact with each other and the order of interactions
  - Collaboration diagram (Shows the relationships between the objects in a system
  - Statechart diagram (This represents the Conditions of the System at a finite instance of time)
  - Activity diagram (represents workflow of System)

Requirements modeling
(It is an essential part of requirements engineering. It involves creating Various models to capture and analyze Software requirements)

- Requirements Modeling Approaches
  - 1) structured Analysis
  - 2) Object - Oriented analysis
- Types of models
  - Scenario based models
  - Data models
  - class oriented models
  - flow oriented models

Requirements Negotiation
(To negotiate the requirements of a System to be developed, it is necessary to identify Conflicts)

- Conflict identification
- Conflict analysis
  - A data conflict
  - conflict of interest
  - conflict of value
  - Relationship conflict
  - structural conflict
- Conflict resolution
- Documentation of the conflict resolution

Requirement Validation

- Quality Aspects of Requirements
  - is based on
    - 1) content
    - 2) Documentation
    - 3) Agreement
- principles of requirement Validation
- Requirements Validation Techniques
- Fundamentals of Requirements Validation

Design within the Context of Software Engineering

- what is design ? (it is a place where creativity rules and technical Considerations all Come together in a formulation)
- Steps of designing
  - Architecture
  - Interface
  - s/w Components
- Design Concepts
- Design process
  - Software Quality
- Design Model

Design Concepts

- Abstraction
  - procedural abstraction
  - data abstraction
- Architecture
  - Structural models
  - framework models
  - Dynamic models
  - process models
  - functional model
- pattern
- Seperation of Concerns
- Modularity
- Information Hiding
  - Hiding
  - Abstraction
- functional Independence
  - cohesion
  - Coupling
- Refinement
- Aspects
- Refactoring
- object oriented design concepts
- design classes

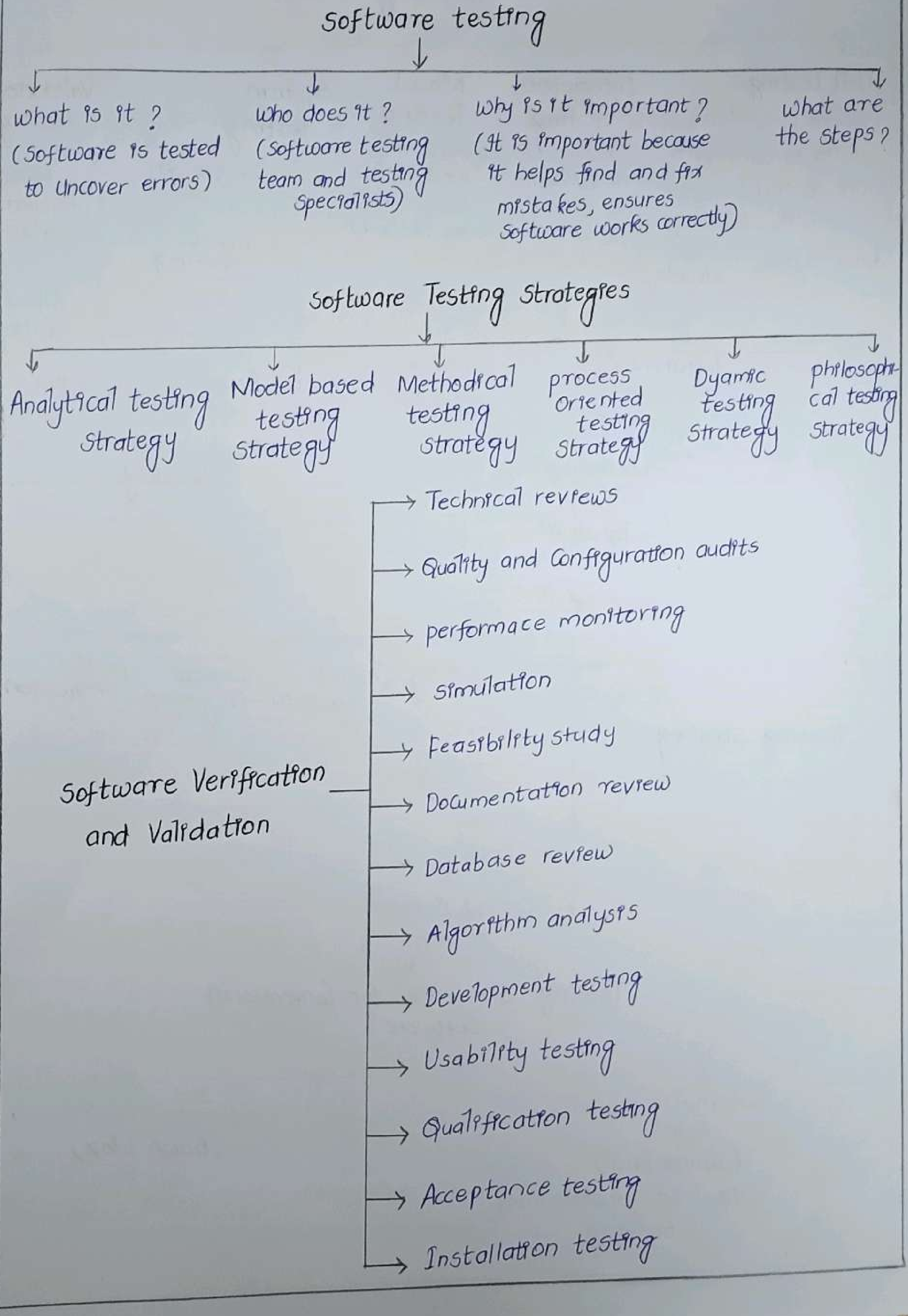
Design Models (can be viewed in two different dimensions)

Here Graph diagram Important

- (Horizontally) The process dimension
- (Vertically) The abstraction dimension

- Data Design Elements
- Architectural design elements
- Interface Design elements
- Component level design elements
- Deployment level design Elements

Module-3 Studychart



Software testing

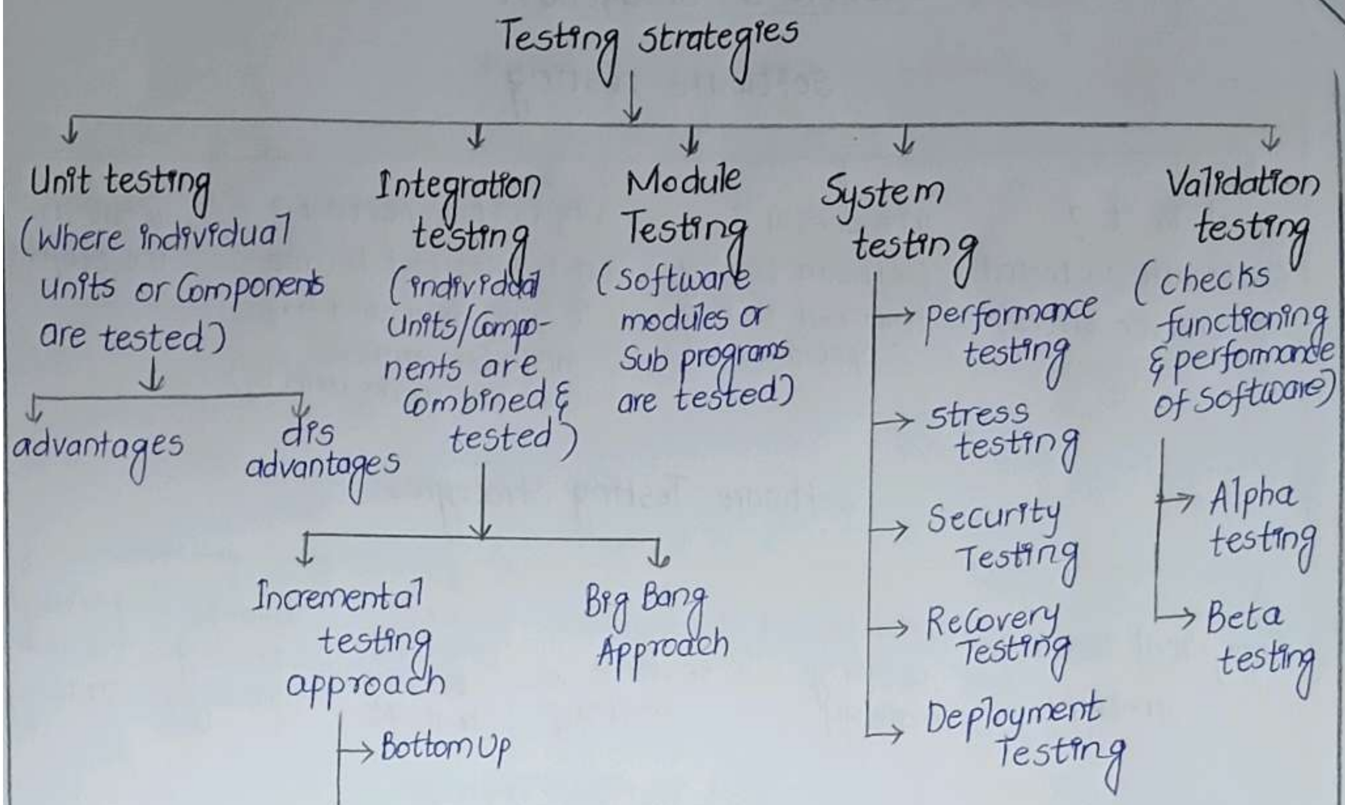
- what is it ? (Software is tested to uncover errors)
- who does it ? (Software testing team and testing specialists)
- why is it important ? (It is important because it helps find and fix mistakes, ensures software works correctly)
- what are the steps ?

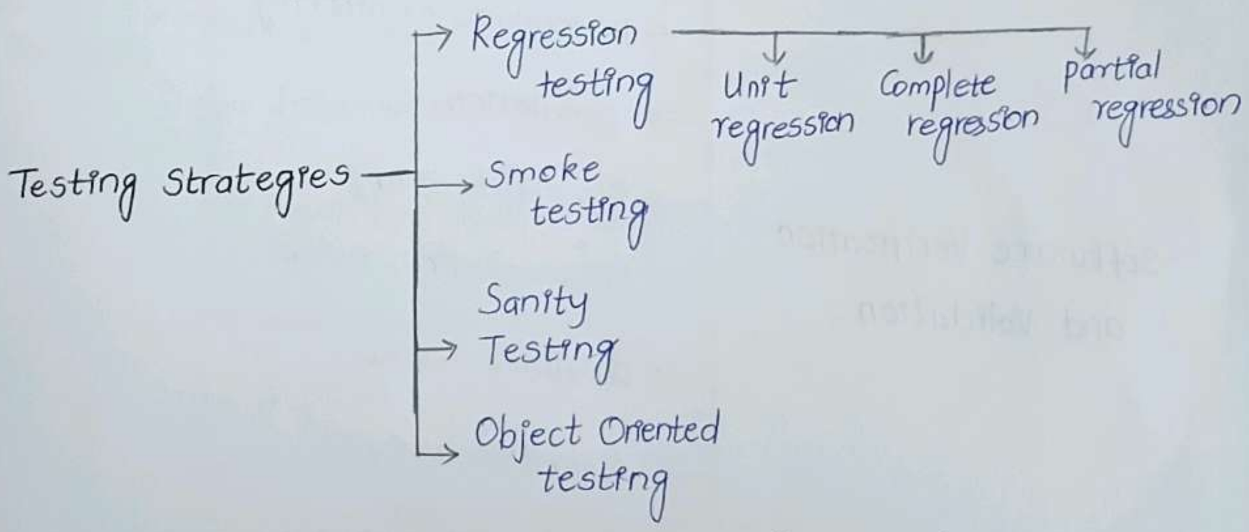
Software Testing Strategies

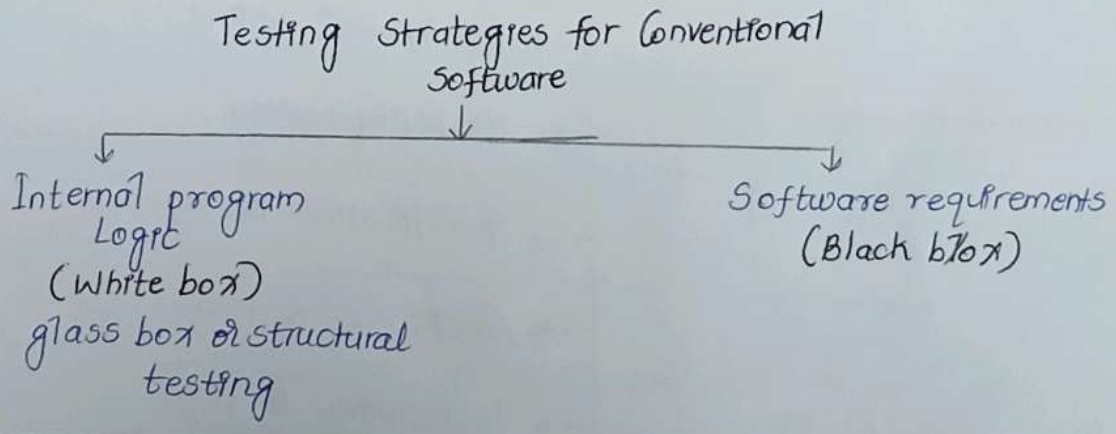
- Analytical testing strategy
- Model based testing strategy
- Methodical testing strategy
- process oriented testing strategy
- Dyamic testing strategy
- philosophical testing strategy

Software Verification and Validation

- Technical reviews
- Quality and Configuration audits
- performace monitoring
- simulation
- Feasibility study
- Documentation review
- Database review
- Algorithm analysis
- Development testing
- Usability testing
- Qualification testing
- Acceptance testing
- Installation testing


Testing strategies
Unit testing (Where individual units or Components are tested)
advantages
dis advantages
Integration testing (individual units/Components are Combined & tested)
Incremental testing approach
Bottom Up
Top-down
Sandwich
Big Bang Approach
Module Testing (Software modules or Sub programs are tested)
System testing
performance testing
stress testing
Security Testing
Recovery Testing
Deployment Testing
Validation testing (checks functioning & performance of software)
Alpha testing
Beta testing


Testing Strategies
Regression testing
Unit regression
Complete regression
partial regression
Smoke testing
Sanity Testing
Object Oriented testing


Testing Strategies for Conventional Software
Internal program Logic (White box) glass box or structural testing
Software requirements (Black box)

Whitebox Testing Techniques

- Control structure Testing
  - Condition Testing
  - Data flow Testing
  - Loop Testing
    - Simple loop
    - Nested loop
    - Concatenated loop
    - Unstructured loop
- Basis path Testing
  1) Construct the Control flow graph for code
  2) Compute the Cyclomatic Complexity of the graph

     Measure - No.of Errors

     Metrics - No.of Errors found per person

     $V(G) = E - N + 2$ (or) $V(G) = P + 1$

     E = edges, N=nodes in Control flow graph

     P = no. of predicate nodes
  3) Identify the Independent paths
  4) Design test cases from independent paths

Blackbox Testing

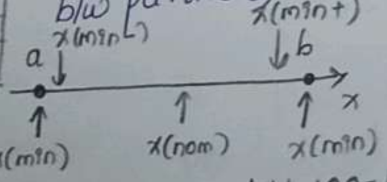
(Behavioral, Opaque-box, Closed box, specification based or eye-to-eye testing)

Important

- Equivalence partitioning (input values to the System or application are divided into different groups)
- Boundary Value Analysis (The process of testing between extreme ends or boundaries b/w partitions)
- Decision Table testing
- State transition testing
- Error Guessing
- Graph based testing
- Comparison testing

class portioning		
invalid	Valid	Invalid
<=17	18-60	>=61

for Age = 18 - 60



for x = 1-1, 1, 1+1, 100-1, 100, 100+1

Module-4 Studychart

product Metrics

(These are developed to check whether a product is developed according to the user requirements)

- Metrics for analysis model (or) Metrics for requirement model
- Metrics for design model
- Metrics for Source code
- Metrics for testing
- Metrics For maintance

Metrics for the Requirements Model

↓ (Here we do project Estimation)

Function-Based Metrics

↓

Function point (FP) metric

(Can be calculated by the estimation of software Information domain)

↓

Information Domain Values

- No. of external outputs (EOs)
- No. of internal logical files (ILFs)
- No. of external interface files (EIFs)
- No. of external inquires (EQs)
- No. of external inputs (EIs)

No. of internal logical files (ILFs) and No. of external interface files (EIFs): Data functional type

No. of external outputs (EOs), No. of external inquires (EQs) and No. of external inputs (EIs): Transactional functional type

$$FP = \underbrace{\text{count total}}_{\text{UFP}} \times \underbrace{[0.65 + 0.01 \times \Sigma(F_i)]}_{\text{VAF (or) CAF}}$$

UFP
↓
Unadjusted functional point
↓
Information domain Values × Weight

VAF (or) CAF
↓
Value (or) Complexity Adjustment Factor
↓
ΣF_i = There are (14) adjustment factors × Type of factor (ex:- Average = 3)

Metres for Specification Quality

↓

$n_r = n_f + n_{nf}$

n_f = no. of functional requirements

n_{nf} = no. of non-functional requirements

↓

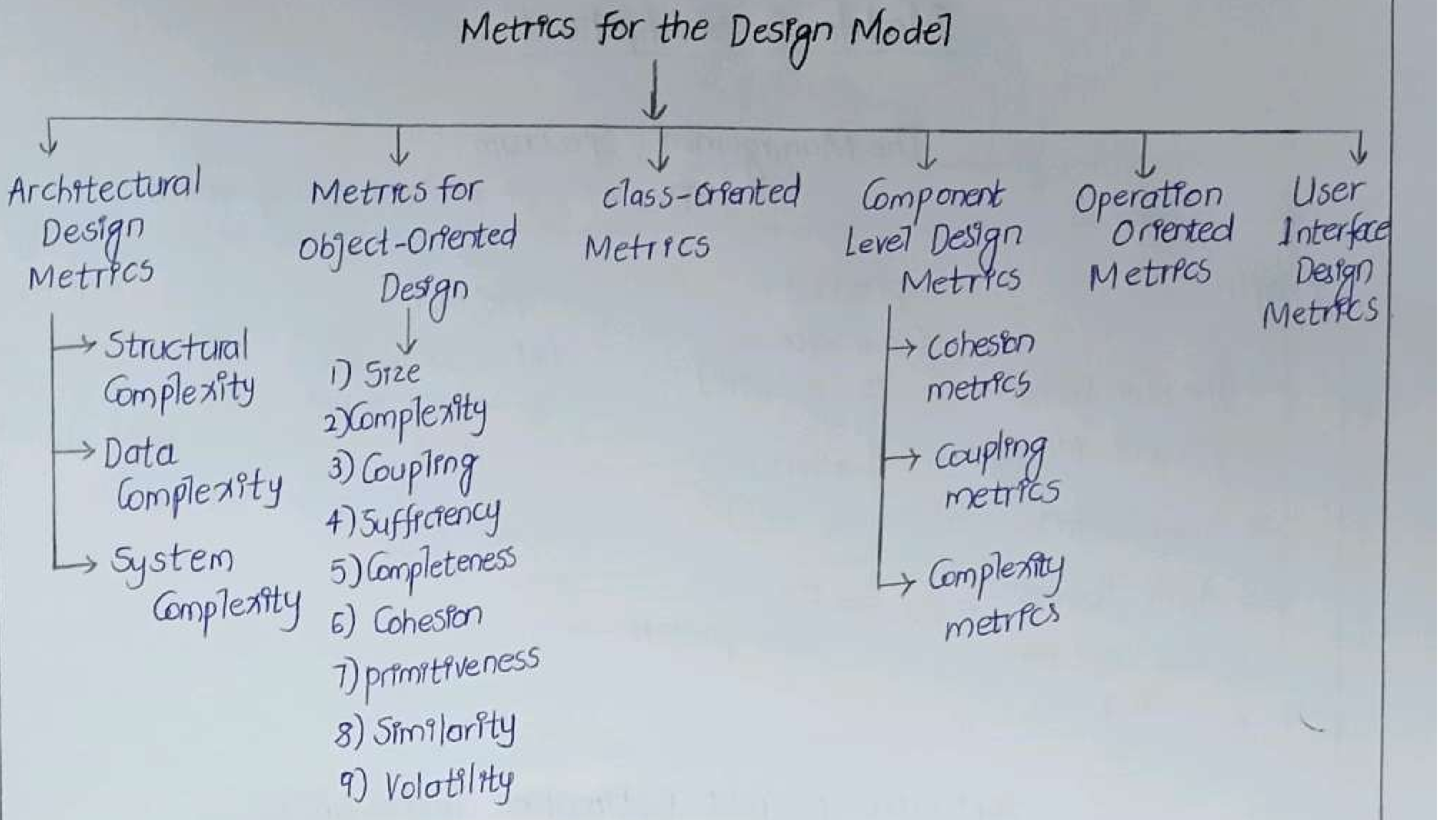
To determine Specificity of requirements

$$Q_1 = \frac{n_{ui}}{n_r}$$

n_{ui} = no. of requirements for which all review are same

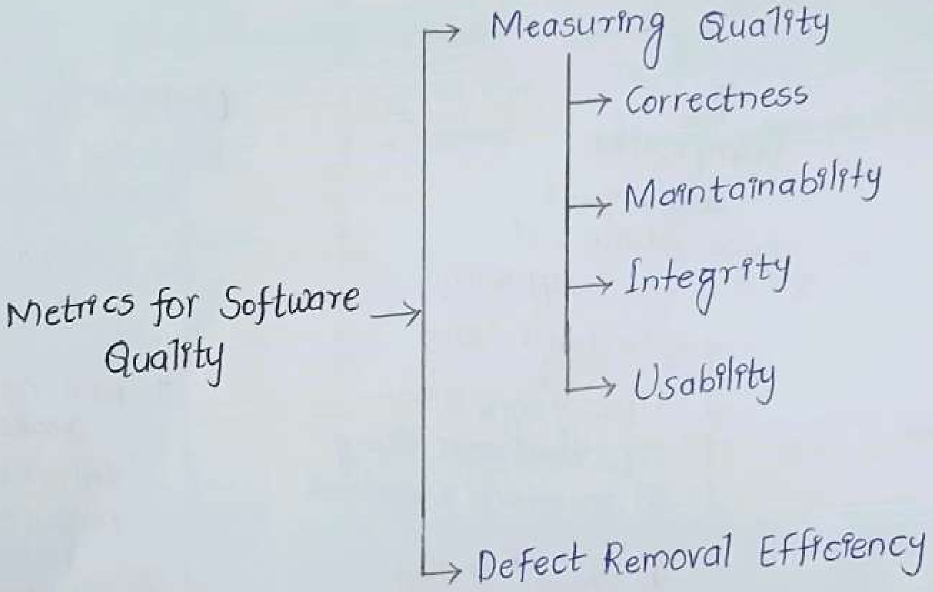
If $Q_1 \simeq 1$ has less ambiguity

$$Q_2 = \frac{n_u}{n_i \times n_s} \qquad Q_3 = \frac{n_c}{n_c + n_{nv}}$$



Metrics for the Design Model

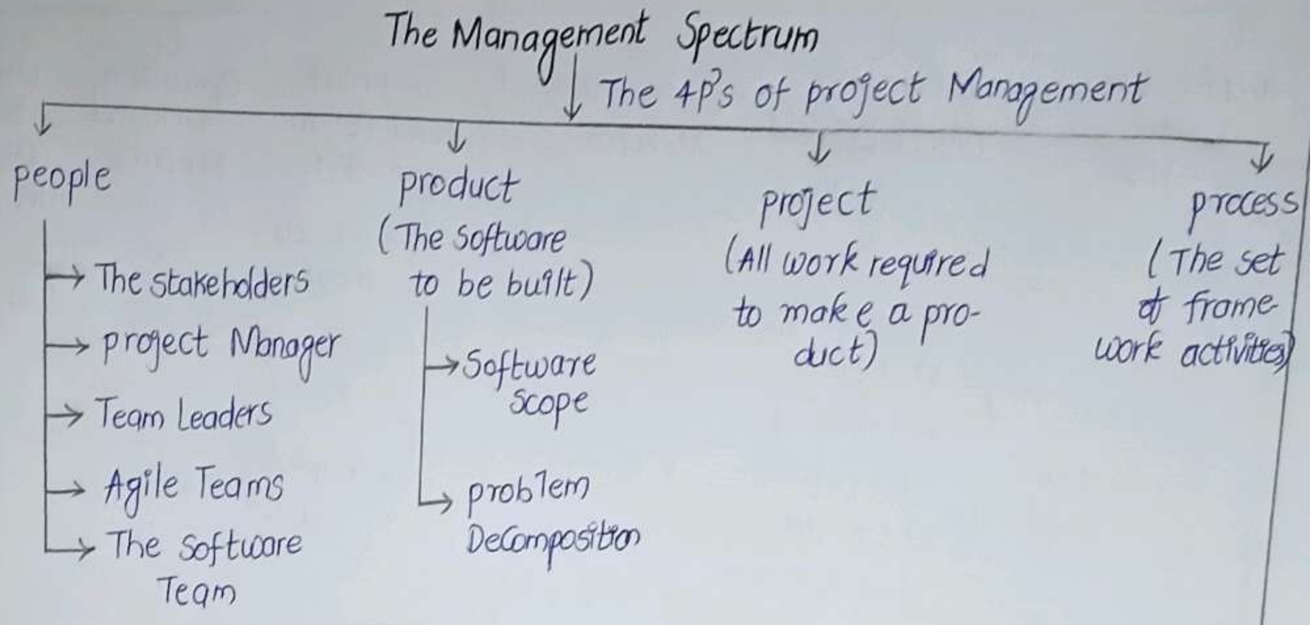
- Architectural Design Metrics
  - Structural Complexity
  - Data Complexity
  - System Complexity
- Metrics for Object-Oriented Design
  1) Size
  2) Complexity
  3) Coupling
  4) Sufficiency
  5) Completeness
  6) Cohesion
  7) primitiveness
  8) Similarity
  9) Volatility
- Class-Oriented Metrics
- Component Level Design Metrics
  - Cohesion metrics
  - Coupling metrics
  - Complexity metrics
- Operation Oriented Metrics
- User Interface Design Metrics



Metrics for Software Quality

- Measuring Quality
  - Correctness
  - Maintainability
  - Integrity
  - Usability
- Defect Removal Efficiency

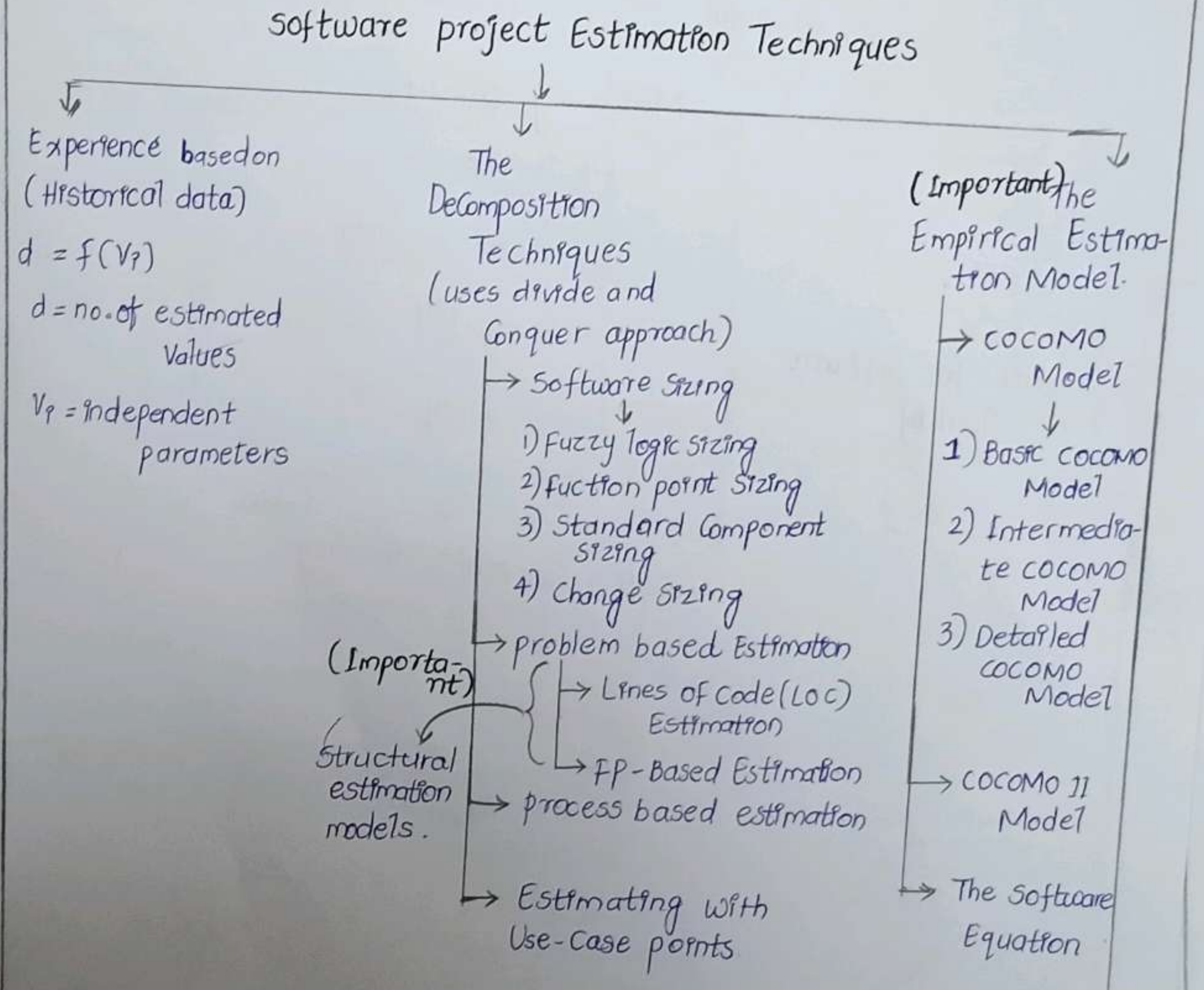
Module-5 Studychart



The Management Spectrum

The 4P's of project Management

- people
  - The stakeholders
  - project Manager
  - Team Leaders
  - Agile Teams
  - The Software Team
- product (The Software to be built)
  - Software Scope
  - problem Decomposition
- project (All work required to make a product)
- process (The set of frame work activities)



software project Estimation Techniques

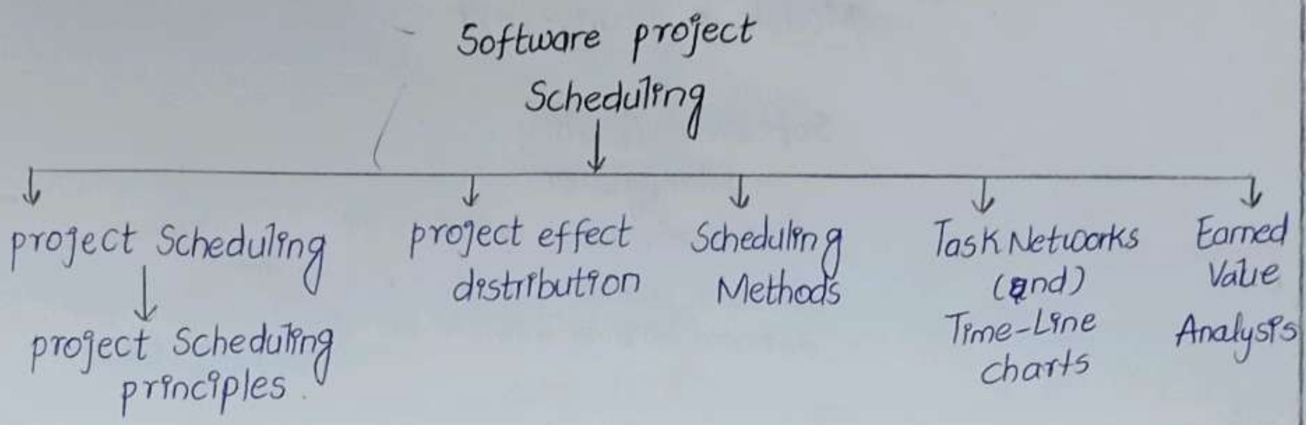
- Experience basedon (Historical data)

  d = f(Vi)

  d = no. of estimated Values

  Vi = independent parameters
- The Decomposition Techniques (uses divide and Conquer approach)
  - Software Sizing
    1) Fuzzy logic sizing
    2) fuction point sizing
    3) Standard Component sizing
    4) Change sizing
  - problem based Estimation (Importa-nt)
    - Lines of code (LOC) Estimation
    - FP-Based Estimation

    Structural estimation models.
  - process based estimation
  - Estimating with Use-Case points
- (Important) The Empirical Estimation Model.
  - COCOMO Model
    1) Basic COCOMO Model
    2) Intermediate COCOMO Model
    3) Detailed COCOMO Model
  - COCOMO II Model
  - The Software Equation



Software project Scheduling

project Scheduling

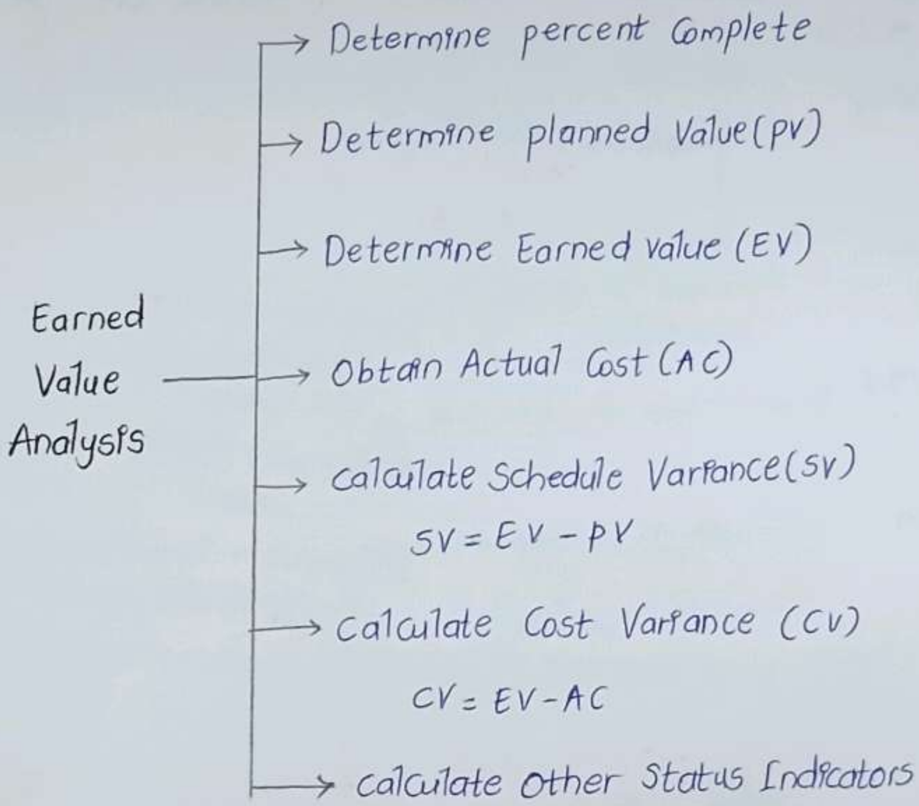
project Scheduling principles.

project effect distribution

Scheduling Methods

Task Networks (and) Time-Line charts

Earned Value Analysis



Earned Value Analysis

Determine percent Complete

Determine planned Value(PV)

Determine Earned value (EV)

Obtain Actual Cost (AC)

calculate Schedule Variance(SV)

$SV = EV - PV$

Calculate Cost Variance (CV)

$CV = EV - AC$

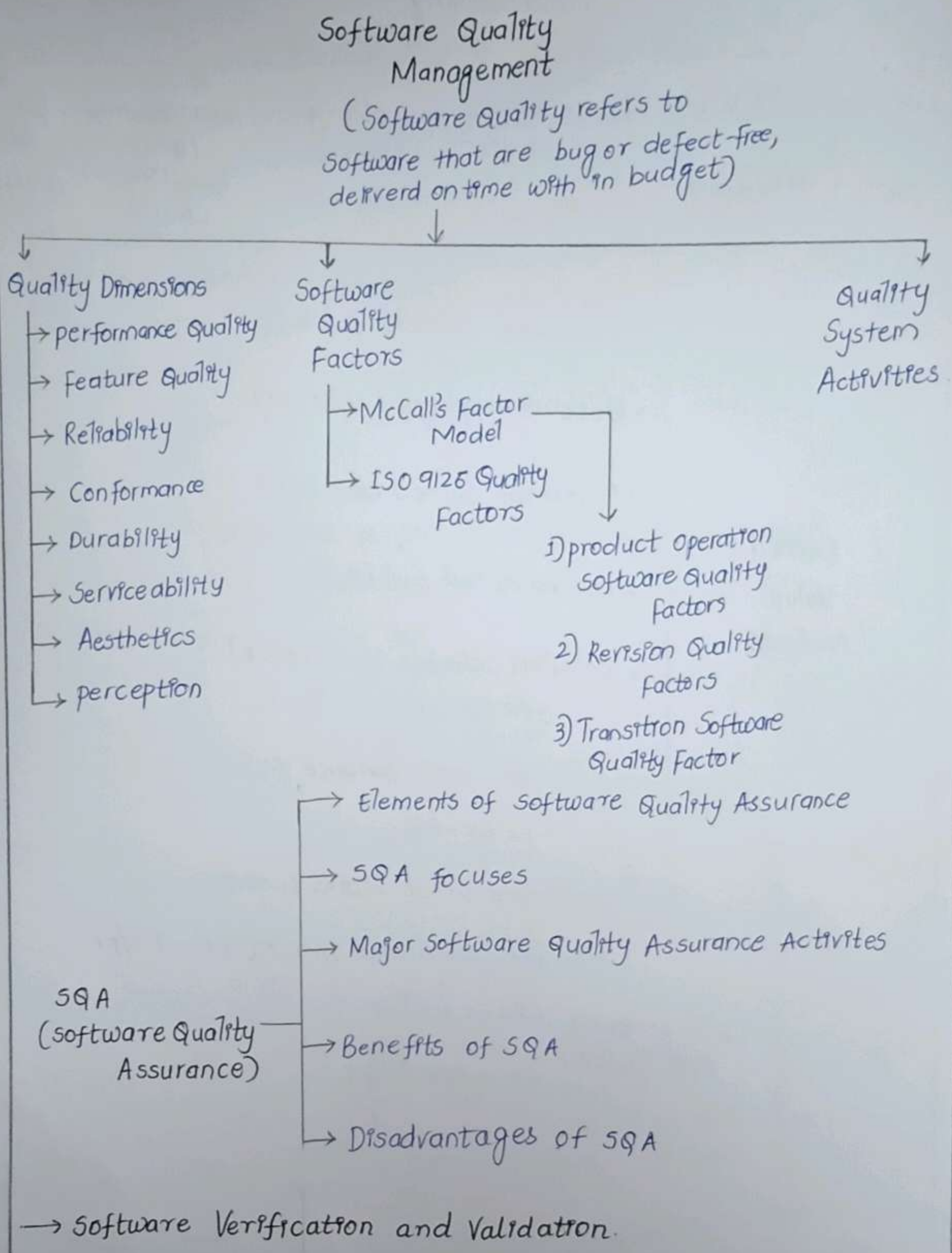
calculate other Status Indicators

Schedule performance Index (SPI) = EV/PV

(Cost performance Index) $CPI = EV/AC$

$CR = SPI * CPI$

Module-6 Studychart



Software Quality Management

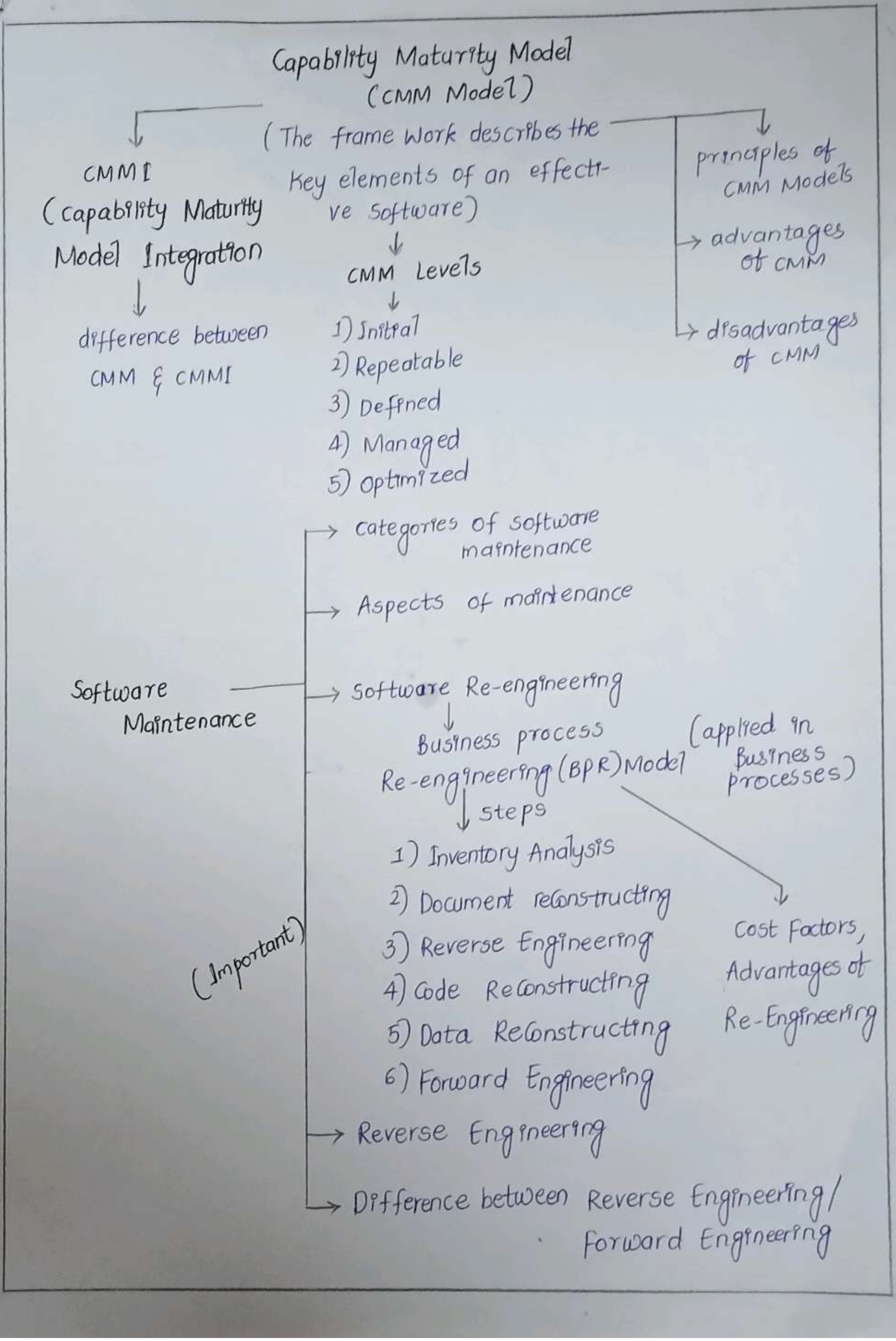
(Software Quality refers to Software that are bug or defect-free, deliverd on time with in budget)

- Quality Dimensions
  - performance Quality
  - Feature Quality
  - Reliabilty
  - Conformance
  - Durability
  - Serviceability
  - Aesthetics
  - perception
- Software Quality Factors
  - McCall's Factor Model
    - 1) product operation software Quality factors
    - 2) Revision Quality factors
    - 3) Transition Software Quality Factor
  - ISO 9126 Quality Factors
- Quality System Activities

SQA (software Quality Assurance)

- Elements of Software Quality Assurance
- SQA focuses
- Major Software quality Assurance Activites
- Benefits of SQA
- Disadvantages of SQA

→ Software Verification and Validation.



Capability Maturity Model
(CMM Model)

(The frame Work describes the Key elements of an effective Software)

CMMI
(Capability Maturity Model Integration

difference between CMM & CMMI

CMM Levels

1) Initial
2) Repeatable
3) Defined
4) Managed
5) optimized

principles of CMM Models

advantages of CMM

disadvantages of CMM

Software Maintenance

(Important)

categories of software maintenance

Aspects of maintenance

Software Re-engineering

Business process Re-engineering (BPR) Model

(applied in Business processes)

steps

1) Inventory Analysis
2) Document reConstructing
3) Reverse Engineering
4) Code ReConstructing
5) Data ReConstructing
6) Forward Engineering

Cost Factors, Advantages of Re-Engineering

Reverse Engineering

Difference between Reverse Engineering/ Forward Engineering