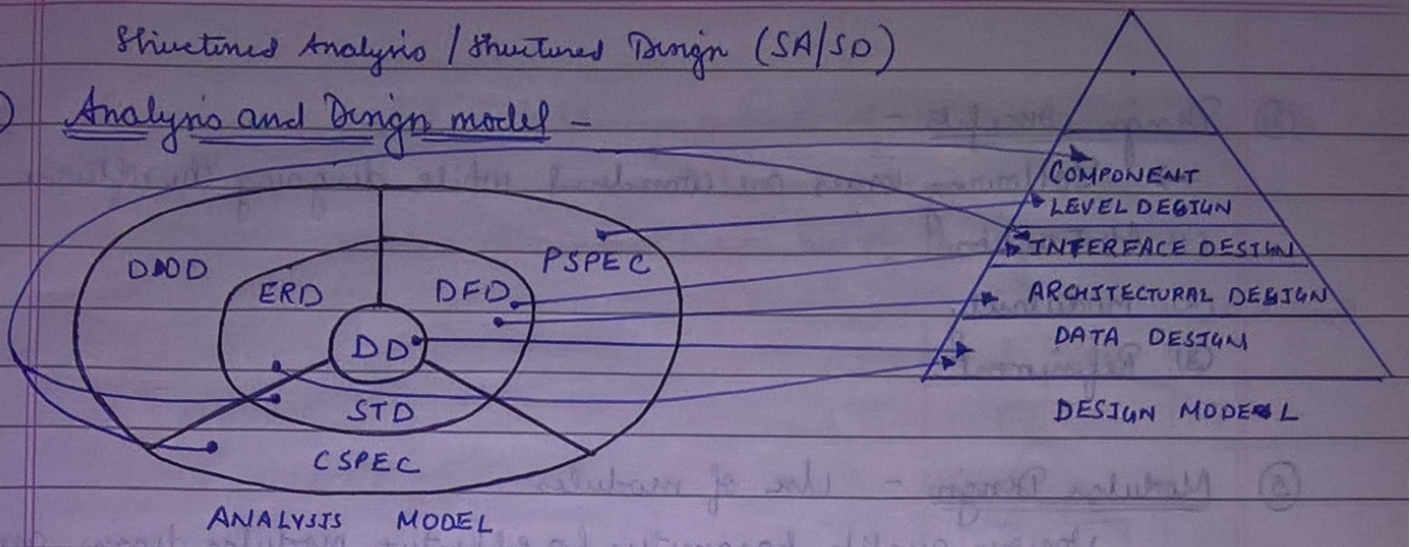
UNIT-3

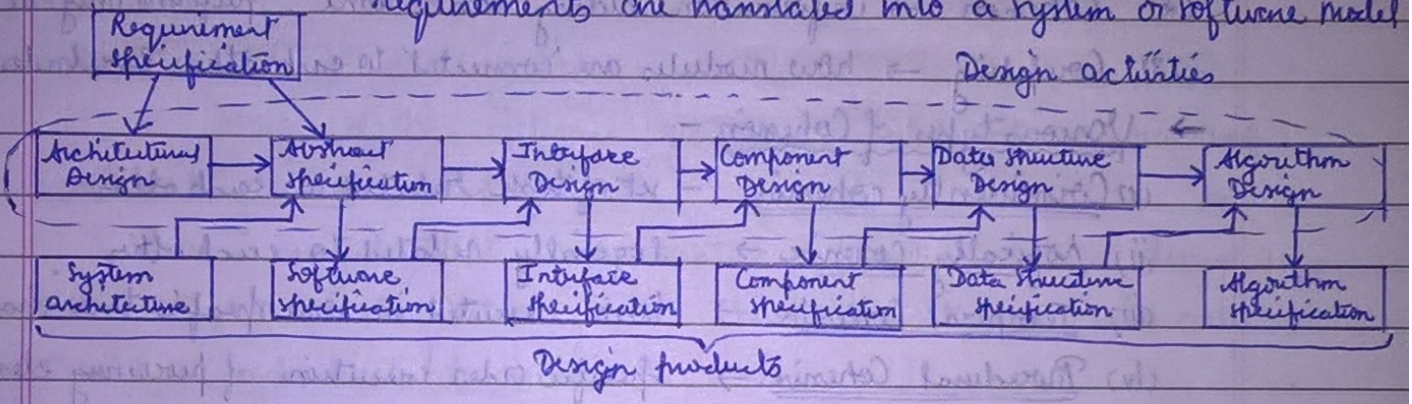
SOFTWARE DESIGN

Structured Analysis / Structured Design (SA/SD)

① **Analysis and Design model** –



② **Design Process** – Sequence of steps carried through which the requirements are translated into a system or software model



③ **Design principles** –

(1) Design process should not suffer from "tunnel vision" ←
(2) Design should be traceable to the analysis model
(3) Design should not reinvent the wheel
(4) Design should "minimize the intellectual distance" between the software and the problem in the real world.
(5) Design should exhibit uniformity and integration ←
(6) Design should be structured to accomodate change ←
(7) Design should be structured to degrade gently ←
(8) Design is not coding and coding is not design
(9) Design should be assessed for quality.
(10) Design should be reviewed to minimize conceptual errors

④ Design concepts -

→ Following issues are considered while dingning the software -

(1) Abstraction

(2) Modularity

(3) Refinement

⑤ Modular Design - Use of modules

Various quality parameters for effective modular design are -

(1) Functional Independence → functional modules with single minded app

(2) Cohesion → information hiding can be done. performs only one thin

(3) Coupling → how modules are 'connected' to each other & outside wo

→ Various types of Cohesion -

(i) Coincidently cohesive → set of task related to each other

(ii) Logically Cohesive → logically related to each other

(iii) Temporal Cohesive → tasks executed in some specific time span

(iv) Procedural Cohesive → specific order execution of processing element

(v) Communicational Cohesive → sharing of data between processing elemen

→ Various types of Coupling -

(i) Data coupling - done by parameter passing & data interaction

(ii) Control coupling - share related control data

(iii) Common coupling - Common data & global data

(iv) Content coupling - Use content of another module.

→ Fan-out - Number of immediate subordinates to a module

→ Fan-in - how many modules directly control a given module

→ Factoring - Separation of function into new modules

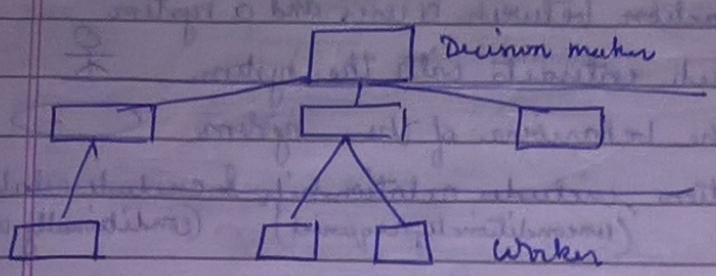
⑥ Software Modelling -

In software architecture the software model is designed and structure of that model is partitioned horizontally or vertically.

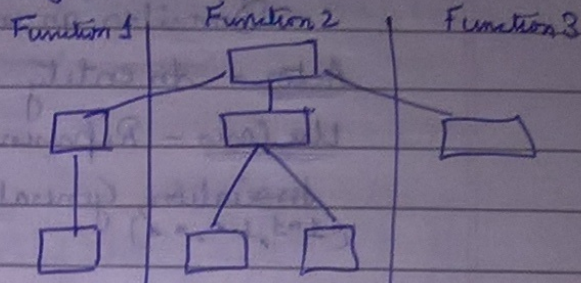
(1) Structural model → organized collection of components

(2) Framework model → Reusable architectural design frameworks

(3) Dynamic model - behaviour aspects of the program architecture
(4) Process model - business or technical process
(5) Functional model - functional hierarchy of the system.



Horizontal Partitioning



Vertical Partitioning

(7) Function oriented design -

Design of the software system is divided into various interacting units called function

Various models that can be created in function oriented design are - (1) DFD, (2) Data Dictionary, (3) Structured charts, (4) Psedo codes.

→ Structured charts / design -

Basic element is module. consists of four attributes are Input and Output, function, mechanics and internal data.

Module = ▭ Connector = → Flag = (Flag) (Data)

→ Psedo code -

It is combination of algorithm written in simple language and programming language statements.

→ Flow chart - Graphical representation of an algorithm.

I/O symbol = ▭ Process = ▭, Flow of information = ⟷

Decision symbol = ◇→ Connector = ○ START STOP

(8) Object-oriented design -

It includes classes, objects, encapsulation, information hiding, inheritance, polymorphism.

Disadvantages - very slow, require more memory.

Advantages - modules, error efficiently identified, reduces maintenance cost reusability.

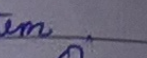
(9) Unified Modelling Language (UML)-

(1) User's View -

→ **Use Case Diagram** -

Describes an interaction between a user and a system.

Actor - An entity which interacts with the system. 

Use Cases - Represent the behaviour of the system 

Association, Generalization, include relationship & exclude relationship
(1-1, 1-*, *-*) (unconditionally required) (conditionally intended)

(2) Structural view -

→ **Class Diagram** - (and object)

Used to capture the static view of the system.

OBJECT NAME
DATA MEMBERS AND ATTRIBUTES
OPERATIONS OR METHODS

(3) Behavioural View -

→ Collaboration Diagram

→ **Sequence Diagram** -

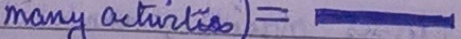
Sequence of events in which how the object interacts with the other object is shown.

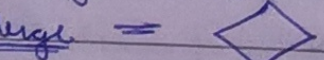
User =  , Active Object =  , life line =  , Active procedure = 

Interaction =  Destruction = 

→ **Activity Diagram** -

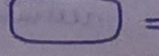
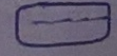
Graphical representation for representing the flow of interaction within specific scenario.

Join & Fork (Represent many activities) =  Start = 

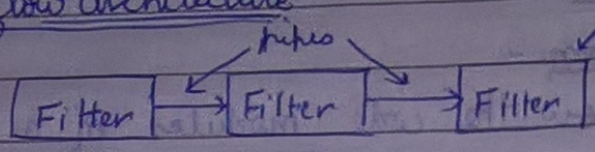
Branch & Merge =  End = 

→ **State transition Diagram** - (Dynamic models)

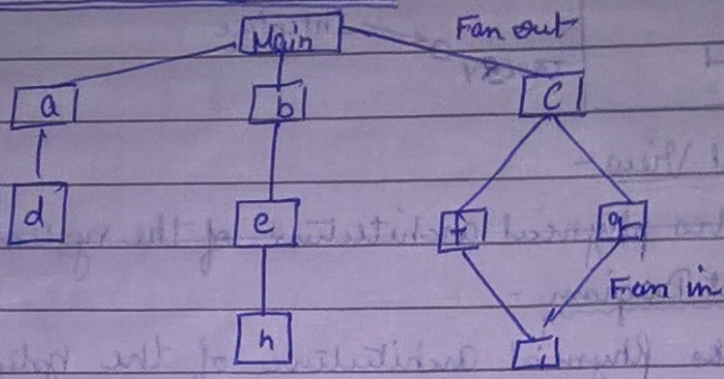
Graphically represent the states of the system. composite state

● = Initial state ⊙ - Final state  = simple state 

(2) Data flow architecture -

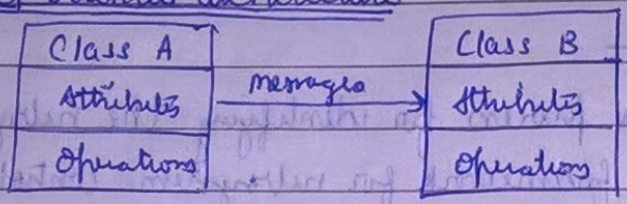


(3) Call and Return Architecture -

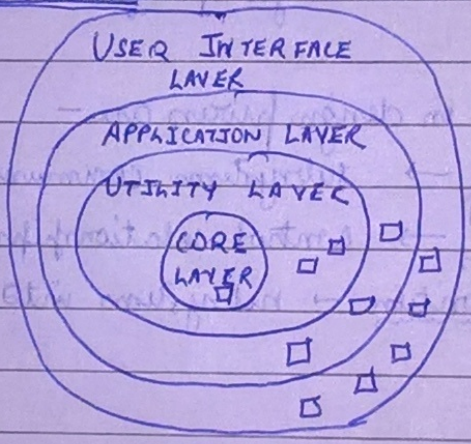


Hierarchical control for call and return is represented.

(4) Object oriented architecture -



(5) Layered architecture -



□ → components

(12) Architectural Views -

(1) Logical view
(2) Process View
(3) Development view
(4) Physical view

(13) User Interface Design -

Effective interaction of system with user is provided.

Eg - GUI (Graphical user interface)

→ User Interface design principles -
(1) User familiarity
(2) Consistency
(3) Minimal surprise
(4) Recoverability
(5) User Guidance

→ Three rules for user interface design (Golden rules) -
(1) Place the user in control
(2) Reduce the user's memory load
(3) Make the interface consistent.

(14) Design Metrics -

→ Architectural Design Metrics -

(1) Metrics by Card and Glass - Three design complexity measure -

Structural complexity → $S(k) = f_{out}^2(k)$; fan out → f_{out}

Data complexity → $D(k) = \frac{tot\_var(k)}{[f_{out}(k)+1]}$; tot_var → total no. of input/output variables

System complexity → $S_y(k) = S(k) + D(k)$ Structural + Data complexity

(2) Metrics by Fenton -
Size = n + e (node + edge)
Depth = Longest path from root to leaf node
Width = Maximum number of nodes at particular level
Edge to node ratio, $r = e/n$

(3) Metrics by using Design structure quality index (DSQI) -
Value of indices are either 0 or 1 (F1 - F7)

→ Object oriented Design metrics -

(1) CK metrics suite - Six class based design metrics are- Weight methods per class (WMC), Depth of inheritance tree (DIT) Number of children (NOC), Coupling between object classes (CBO), Response for a class (RFC), Lack of cohesion in methods (LCOM)

(2) MOOD Metrics suite - Two metrics.
method inheritance factor (MIF), Coupling factor (CF).

(3) Lorenz and Kidd OO metrics -
Class metrics, Inheritance metrics, Class internals, Internals, operation-oriented metrics

→ Component level design metrics - Three means/metrics are - Cohesion, Coupling and complexity.

→ User interface design metrics -

(1) Layout appropriateness -

$$\text{Cost} = \sum_{\text{All transitions in layout}} [\text{frequency of transition} * \text{cost of transition}]$$

(2) Cohesion Metric -

$$\text{Weak function cohesion} = \frac{\text{Number of glue tokens}}{\text{Total no. of data tokens}}$$

$$\text{Strong function cohesion} = \frac{\text{Number of glue tokens}}{\text{Total 3 of data tokens}}$$