Date .....

UNIT-3

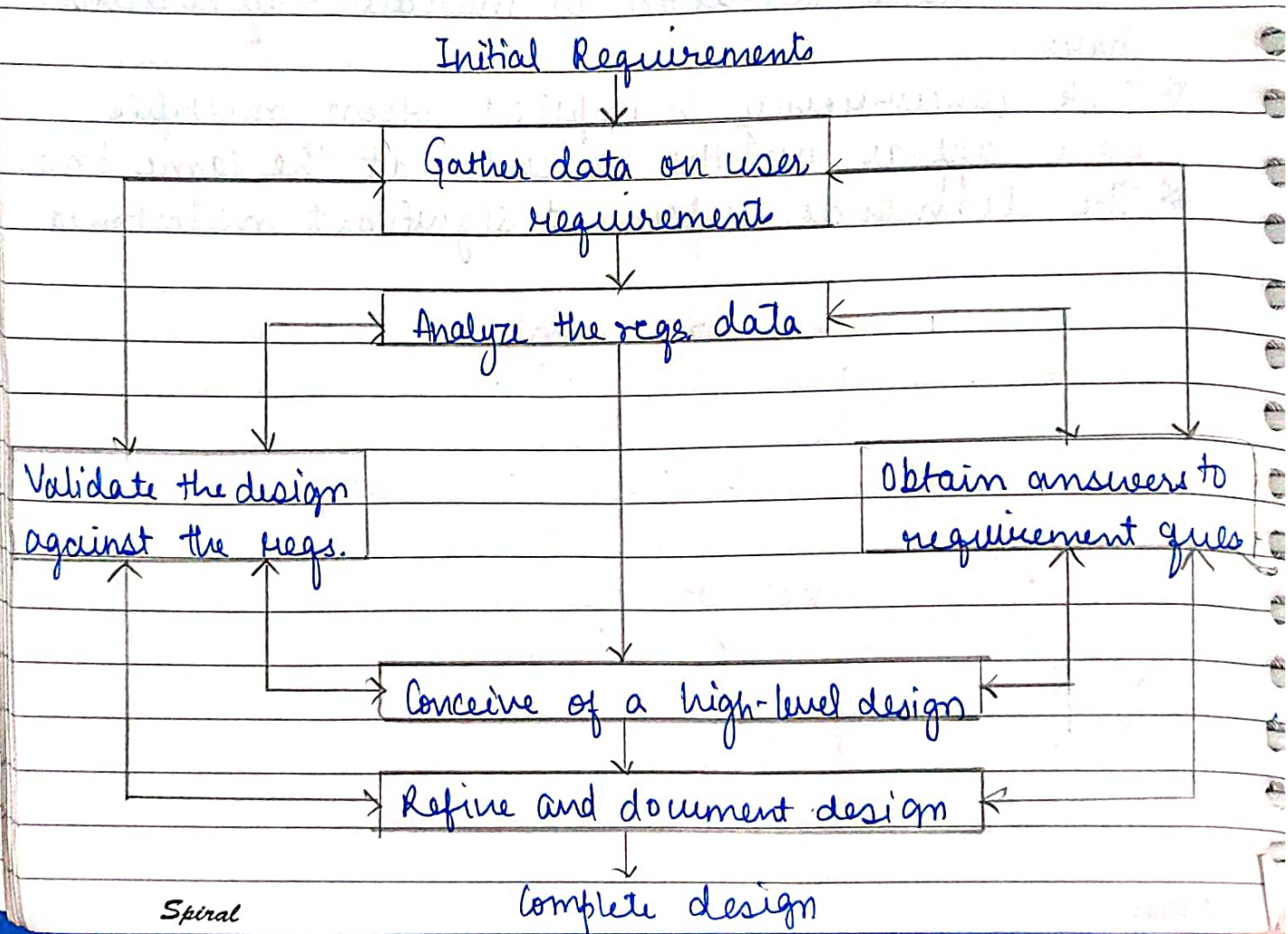
Design Modeling

Software Design

* Software design plans how a software system should be produced to make it functional, reliable, modify and maintain.
* SRS tells what a system does and software design tells how a system works.
* Designing software means determining how requirements are realized and result is a software design document (SDD).

Framework of the Design.

* It starts with Initial requirements ends up with the final design.



Date .....

Conceptual and Technical Designs

Design → Conceptual Design
Design → Technical Design
} Iterative process.

Conceptual Design	Technical Design
• For the customers.	• For the system builders.
• Tells the customer what the system will do.	• Allows the system builders to understand the actual h/w and s/w needed to solve the customer's problem.
• After approval from user transferred to technical design	
• Describes the system in a way understandable to customer.	• Describes the h/w configuration and s/w needs, communication iterface, etc.
• Independent of implementation	• Translates the requirement into a solution.

Objectives of Design (Good Design)

1. Correct and complete.
2. Understandable.
3. Maintainable, and to facilitate maintenance of the produced code.
4. At the right level.

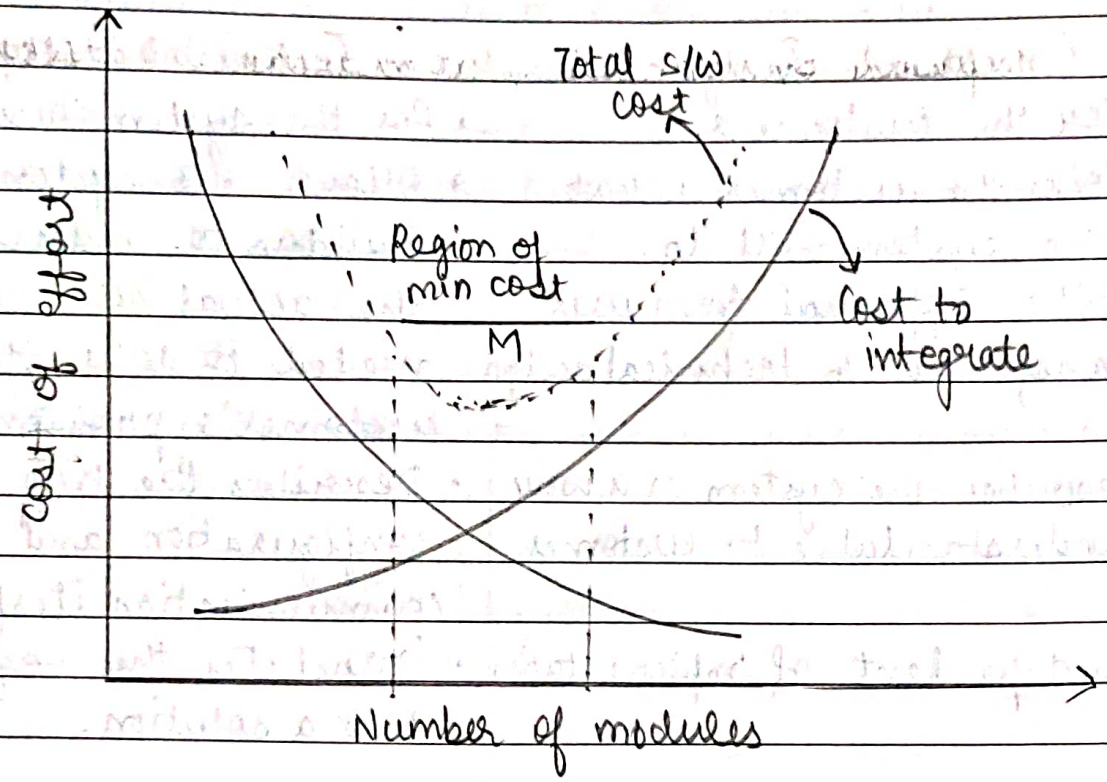
Modular Design (Modularity)

* A modular system consist of well-defined manageable units with well-defined interfaces.

Date .....

* It uses the divide and conquer approach.
* Modularity enhances design clarity, which eases implementation, debugging, testing, documenting and maintainance.

Affect of module design on software cost.



Modularity and Software Cost.

As, no. of modules ↑, effort associated with integrating the module ↑.

There is a number M, of modules that would result in minimum development cost.

Date .....

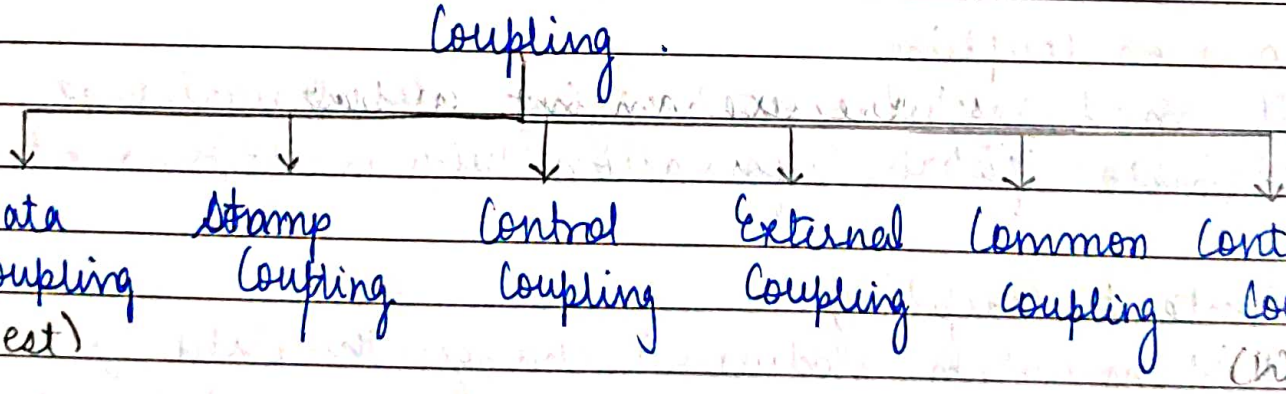
Module Coupling

* Coupling is the measure of degree of interdependence between modules.
* Low coupling are not dependent →
* It is measured by the number of interconnections between modules.
* * A good design will have low coupling.
(Changing one module will require less effort).

Factors on which coupling depends.

1. No. of parameters passed among the modules
2. Avoid passing undesirable data to the calling module to keep the coupling low.
3. Maintain parent-child relationship between calling module and called module.
4. Pass data not the control information.

Types of Coupling.



Data Coupling

- It happens when exchanging data.
- In data coupling, the components are independent to each other.
- Example: Retreiving student address using student roll no.

Date .....

Stamp Coupling

- If two modules exchange complete data structure
- It involves stamp data.
- Example: A print module that accepts some entity and retreive info to construct a message.

Control Coupling

- If control information (flag) is passed.
- Example: A module that retreives either mother's name or father's name depending on the value of a flag.

External Coupling

- If they are dependent on external hardware or external software.
- Related to the communication of external tools and devices.
- Example: Operating system functions.

Common Coupling

- If modules are exchanging global variables.
- Example: Global information with multiple modules.

Content coupling.

- Happens when module A changes the data of module B. If control is passed from A to the middle of module B.

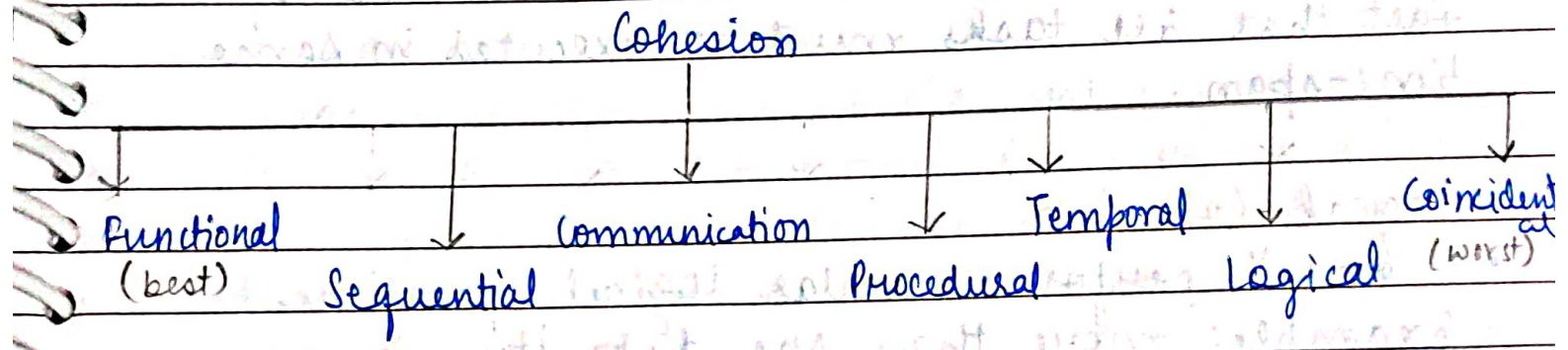
Date .....

Cohesion

* Cohesion is a measure of the degree to which the elements of the module are functionally related.
* Strongly cohesive module implements the functionality that is related to one feature of the solution and requires little or no interaction with other modules.
* High cohesion is desired.

Types of Cohesion

Given a procedure that carries out 2 operations X and Y.



Functional Cohesion

- When X and Y are of a single functional task.
- Example : Calculation of the salary.

Sequential Cohesion

- When X outputs some data that acts as input to Y.
- Example : Addition of marks of individual subjects is used to calculate the GPA.

Communicational Cohesion

- X and Y both operate on same input or contribute to same output.

Spiral

Date .....

Procedural Cohesion

- Occurs in the modules whose instructions accomplish different tasks but yet have been combined and exist an order in which it needs to be finished.
- Example: The report module of an exam system includes: "calculate the GPA, print student result, calculate cumulative GPA, print cumulative GPA," is a procedural cohesion.

Temporal Cohesion

- X and Y must perform around the same time.
- When it contain tasks that are related by the fact that all tasks must be executed in same time-span.

Logical Cohesion

- X and Y perform similar logical operations.
- Example: more than one data item in an input transaction may be a date. Separate code will check for validity of the date. Hence, DATECHECK module can be created for ease of use.

Coincidental Cohesion

- No conceptual relation other than the shared code.
- Example: check validity and print is a single component with 2 parts.

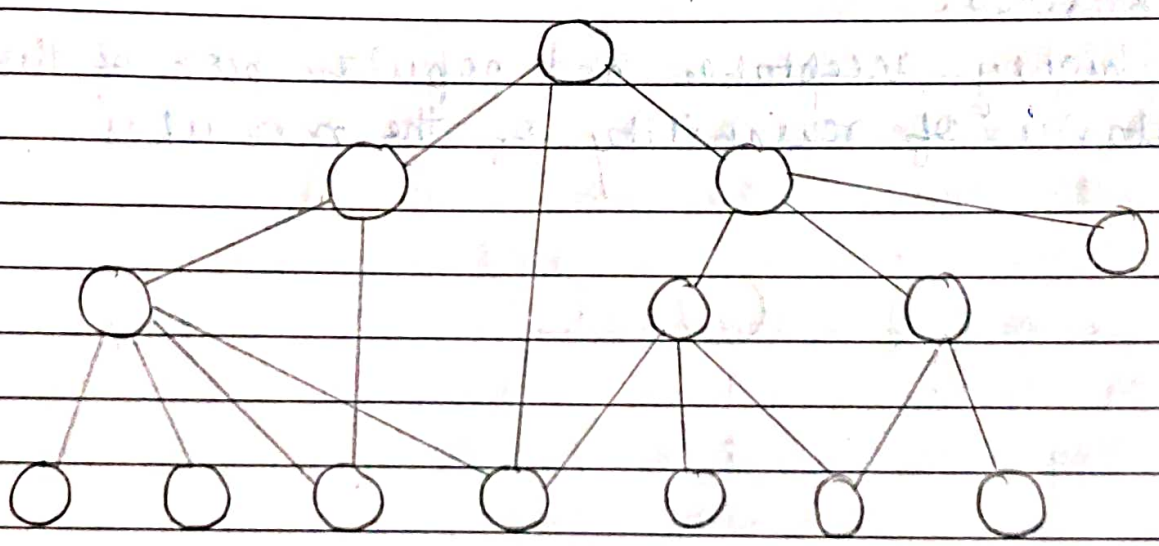
Date .....

Strategy of Design

* A good system design strategy is to organise the program modules in a way that are easy to develop and later to, change.
* Strategies or techniques for performing system design are:

1. Bottom-Up design

- Design progresses from bottom layer upwards, this method is called bottom-up design.
- This approach leads to a style of design where we decide how to combine these modules to provide a larger one till we get the whole of the desired program.
- These modules may be for match functions, for input-output functions, for graphical funs, etc.
- Weakness — we need to use alot of intuition to decide exactly what functionality a module should provide.

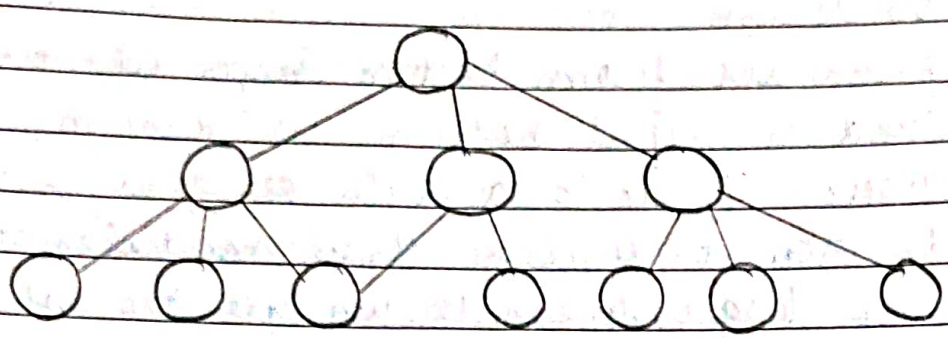


Bottom-up tree structure

Date .....

Top-Down Design

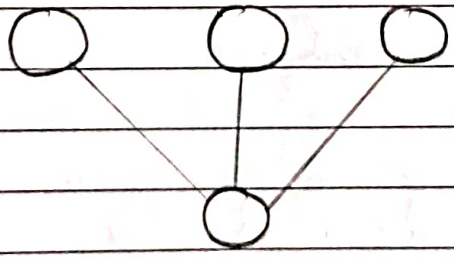
- A top-down design approach starts by identifying the major modules of the system, decomposing them into their lower level modules. until desired level of detail is acheived.
- This is suitable if the specifications are clear and development is from the scratch.



Top-down structure

Hybrid Design

- Pure bottom-up or pure top-down design are often not practical. Hence, hybrid design are implemented.
- It is highly acceptable and popular bcoz of the acceptance of reusability of the modules.



Design reusable structure

Date .....

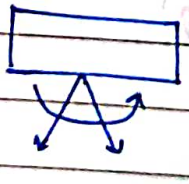
Function Oriented Design

* Function oriented design is an approach to software design where the design is decomposed into a set of interacting units where each unit has a clearly defined function.
Thus, system is designed from a functional viewpoint
* This design divides the whole system into smaller functions, these functional modules can share information among themselves.

Object-Oriented Design

* It works around entities and their characteristics instead of functions involved in the software system.
* It focuses on the entities and their characteristics

Structure Chart

* Structure chart is the hierarchical organization of modules.
* It partitions a system into black boxes.
Black box means the functionality is known to user without knowing the internal design.
* Each program module is represented by a rectangular box.
- Connections are represented by lines.
- → Control
- → Data
-  Conditional Call of module
-  Repetetive call of module.