UNIT-3 Software Design

- Basic concepts of S/w design. (3.1)
- Design strategies
- S/w Measurement & Metrics.

S/w Design:

* It is the process of defining S/w methods, functions, objects and overall structure of the defined S/w.
* The output of S/w design process is design documentation, pseudo codes, detailed logic diagrams, process diagrams and detailed description of all functional and non-functional requirements.

Software Design Levels:

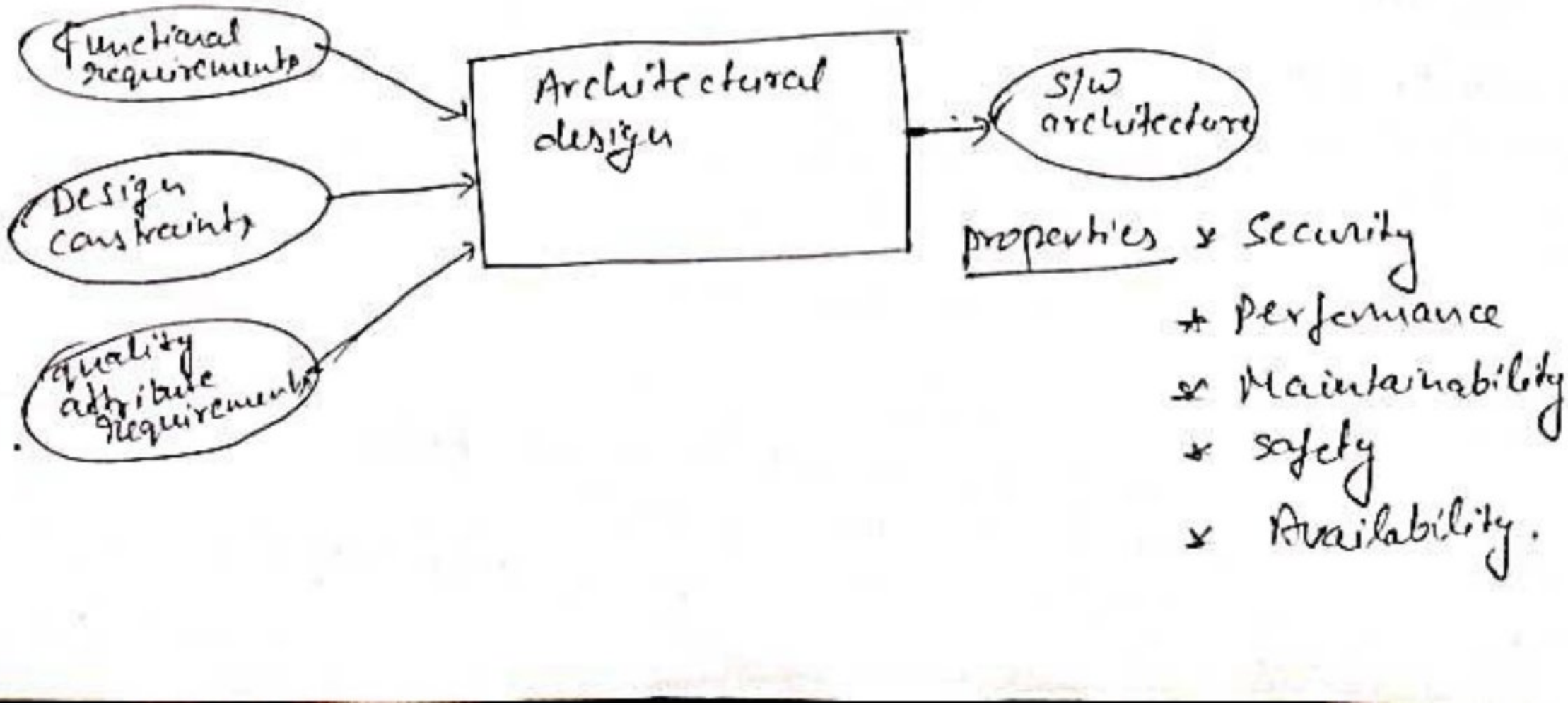
Generally it consists of 3-levels of S/w Design.

1- Architectural Design
2- High Level Design
3- Low-level design.

1. Architecture Design:

✓ IEEE defines architectural design as "the process of defining a collection of H/w and S/w components and their interfaces to establish the framework for the development of computer system."

* It identifies the components that are necessary for developing a computer system and also defines the relationship that exists among the components.



Advantages of architectural Design:

* Works as a tool for stakeholder communication.
* Used for system analysis.
* Facilitates large scale reuse.

High level Design:- / Macro level design:-

* High level design is more specific than architectural design.
* High level design is about starting to make the architecture real. i.e. the real designing starts in high level design.
* High level design consists of :-
  - Logically defining components of the system, and roles and responsibilities of the system.
  - Defining the technology that will be used.
  - Defining the processes that will be used.

Low level Design:- (LLD) / Detailed Design :-

* Low level design is a component-level design process that follows a step-by-step refinement process.
* It describes each and every module in detail i.e. it includes actual logic for each and every component and it goes deep into each modules specification.
* Also k/a micro level / detailed design.
* It converts high level solution into detailed solution.

Concepts used in ~~low level~~ design :

⇒ Modularization ⇒ Cohesion & Coupling. ⇒ flow charts.
⇒ structure charts ⇒ Pseudo codes

Modularization :-

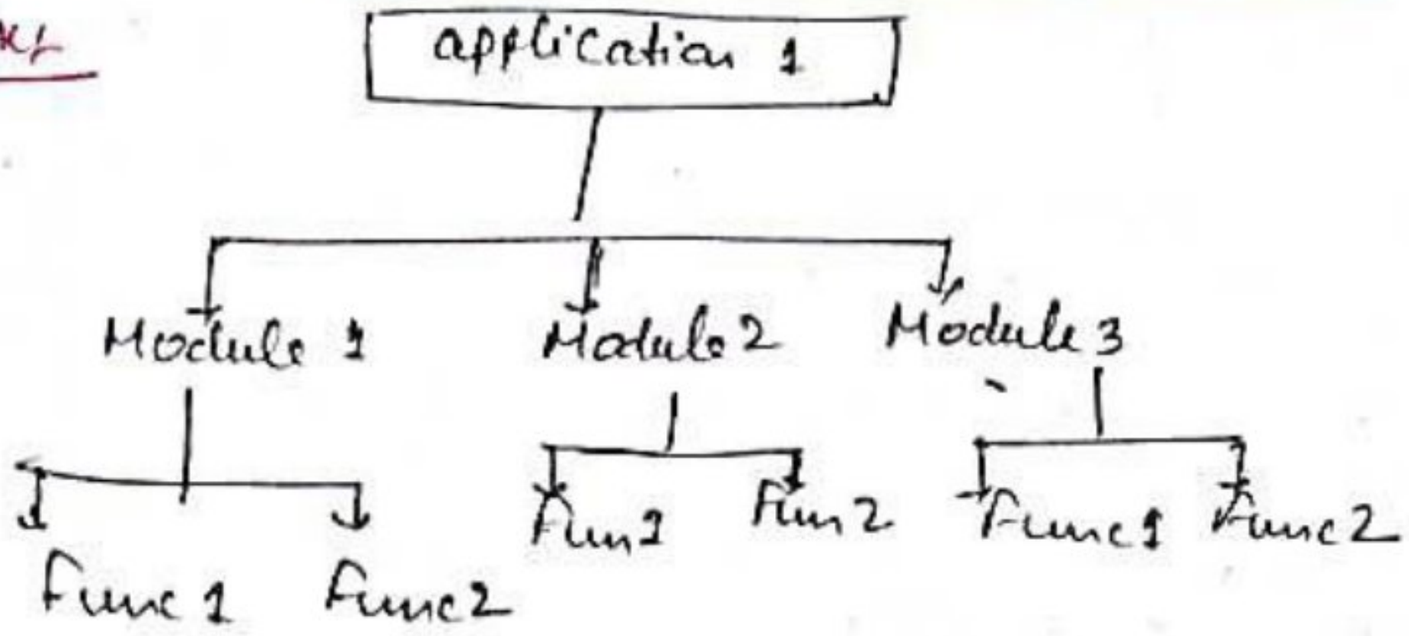
Modularization is a technique in which S/W system is divided into multiple discrete and independent modules, which are expected to be carrying out tasks independently.

These modules works as the basic constructs for the entire software.

Advantages:-

* Smaller components are easier to maintain.
* Easy to understand the S/W system.
* A module can be reused again and again as per requirement

ex:-



Cohesion and Coupling: A system is composed of different modules. So the terms cohesion and coupling are related to the interconnection b/w different parts/ modules of a system.

Cohesion:

It is defined as the degree to which the elements of a particular module are functionally related.

Basically cohesion is used to measure the functional strength of a module. "A good s/w design will have high cohesion."

Types of Cohesion:

High cohesion.

- Functional — elements of modules Modules are functionally related.
- sequential — output of one element is input to other elements.
- Communicational — two elements operate on the same input data or contribute towards same output data. ex:- record updation.
- Procedural — elements ensure the order of execution. ex:- calculating CGPA, print student record.
- Temporal — elements are related by time involved.
- Logical — elements are logically related not functionally. ex:- error handling.
- Coincidental — elements are not related. They can share the location but not work.

Low cohesion.

Coupling:

Coupling is the measure of degree of interdependence b/w the modules.

It measures the strength of relationship b/w modules.

A good software will always have low coupling.

Types:

Tramp data: Data that are passed from one func to another.

Best

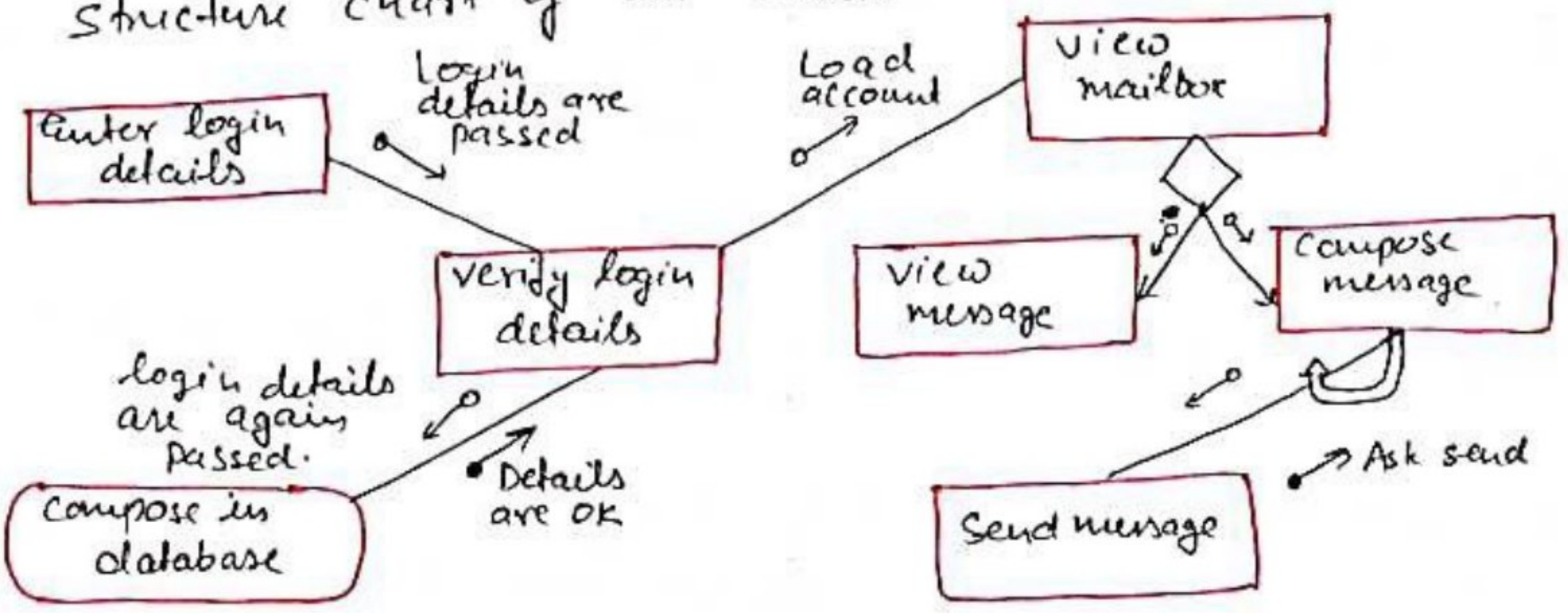
- Data Coupling → Modules communicates by passing data only, modules are independent of each other. and does not contain tramp data. ex: customer billing.
- Stamp " → In this complete data structure is passed from one module to another. It involves tramp data.
- Control " → If modules communicate by passing control eg: ex: sort function.
- External " → Module depends on other module that is external to the s/w being developed.
- Common coupling → The modules have shared some global data. So changes in the global data need to change all the modules that have used that data.
- Content coupling. → One module can modify the data of another module, or control flow is passed from one module to another.

Worst

Design structure Charts:-

* A structure chart (SC) shows the breakdown of a system to its lowest manageable levels.
* They are used in structured programming to arrange programs modules into a tree. Each module is represented by a box, which contains modules name
* The tree shows the relationship b/w modules, showing data transfer b/w the modules.
* Structure charts are example of top-down design.

ex: Structure chart of an email server:

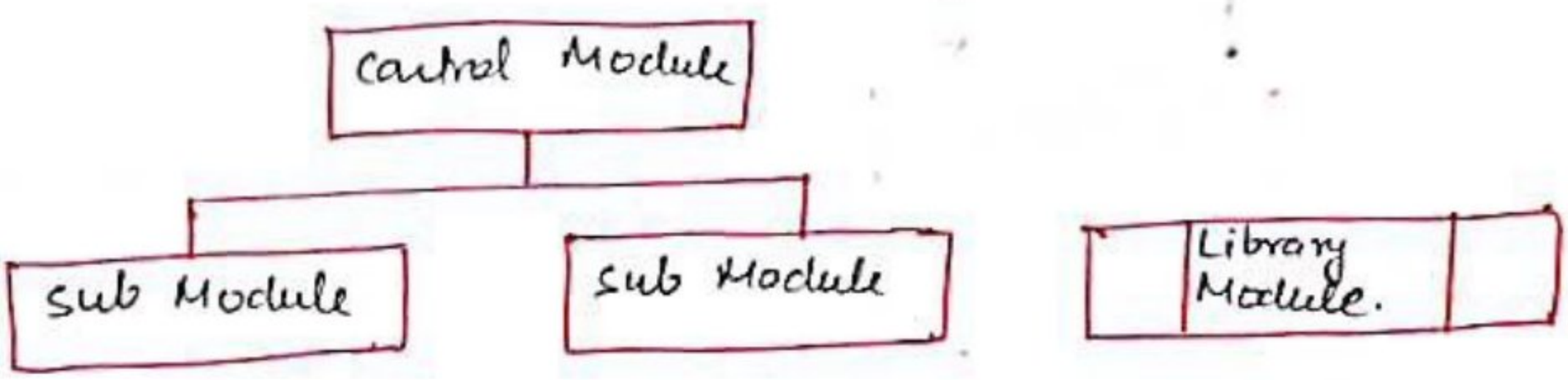


Symbols used in Structured Chart

1) Module:

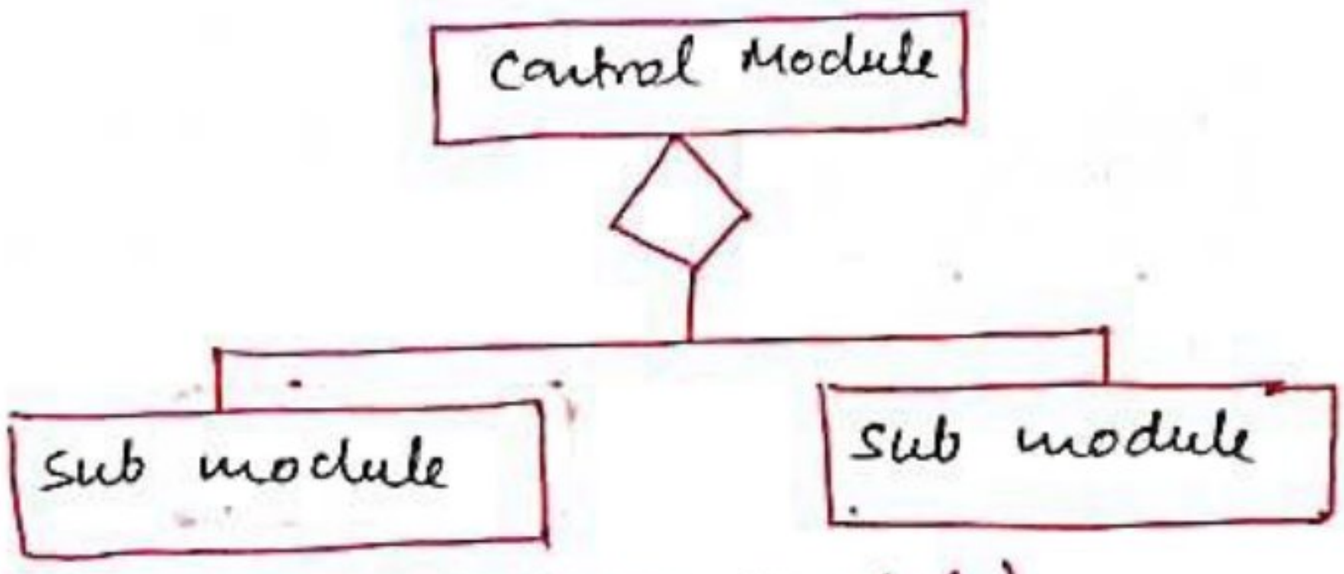
It represents the process or task of the system.
It is of three types.

- **Control Module:**
  A control module branches to more than one sub module.
- **Sub Module:**
  Sub modules are the parts (child) of another module.
- **Library Modules:**
  Library Modules are reusable and invokable from any module.



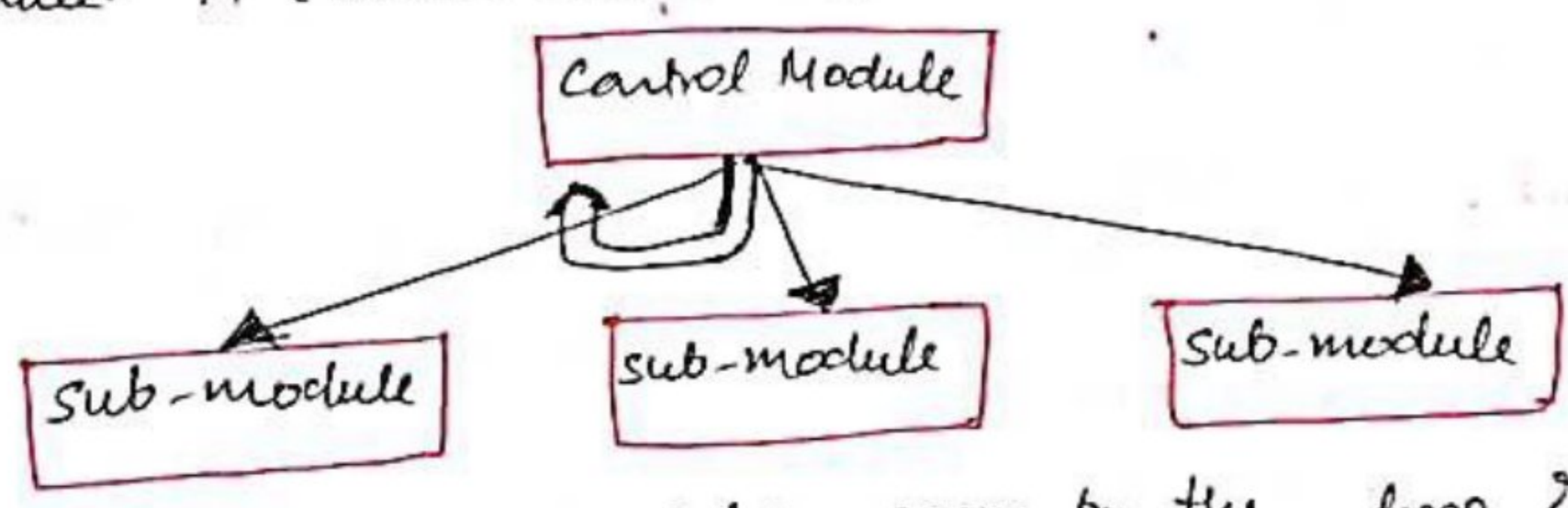
2) Conditional Call:

It represents that control module can select any of the sub module on the basis of some condition.



3) Loop (Repetitive Call of Module)

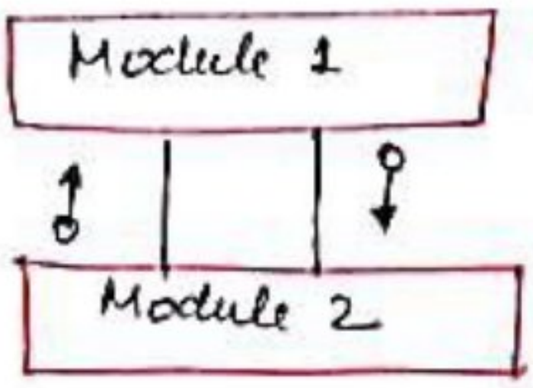
It represents the repetitive execution of module by sub module. A curved arrow represents loop in the module.



Note: All the sub modules cover by the loop repeat execution of the module.

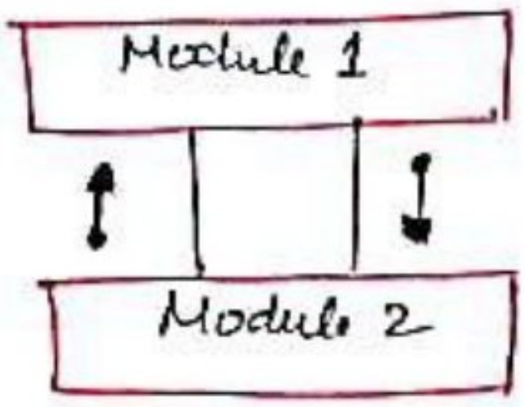
4) Data Flow:-

It represents the flow of data b/w modules. It is represented by directed arrow with empty circle at the end. (2b)

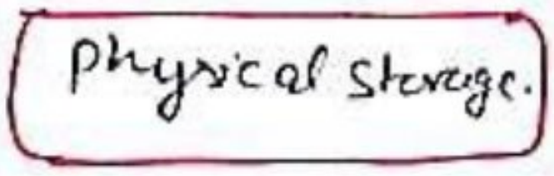


5) Control Flow:

It represents flow of control b/w modules. It is represented by directed arrow with filled circle at the end.

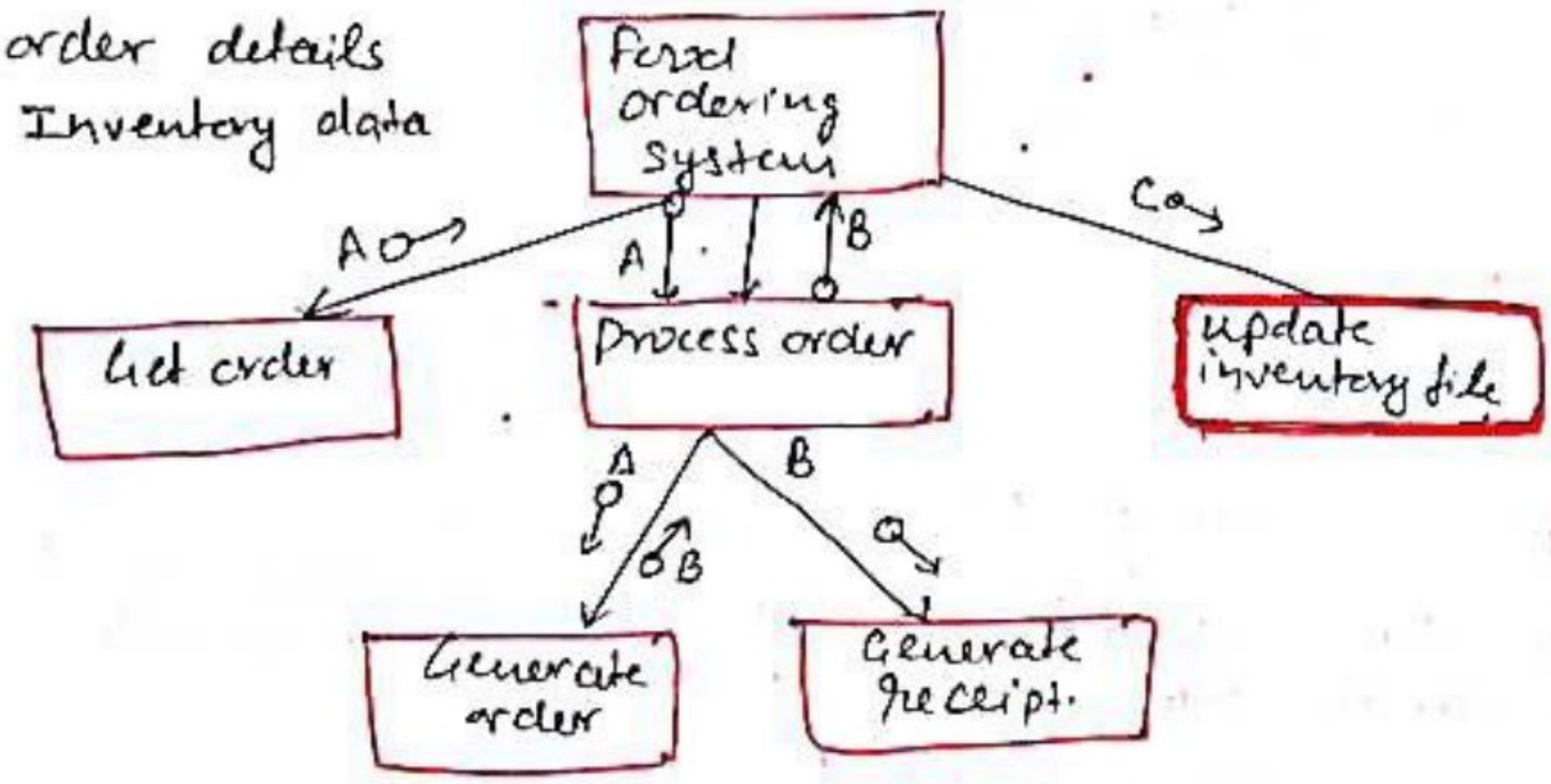


6) Physical storage: where all the infor are to be stored.



Ex: Structure chart of food ordering system.

A:- customer order
B:- order details
C:- Inventory data



Steps: 1. identify modules
2. check for conditionals and loops. If this exists draw the conditions and loops.
3. show the data flow and control flow.
4. Represent the physical storage of the system.

Flow-Charts

* A flowchart can be defined as a diagramatic representation of algorithm, a step by-step approach to solving a task.
* It uses different types of symbols and connect them in order using arrows.
* The diagramatic representation depicts a solution model to a given problem.

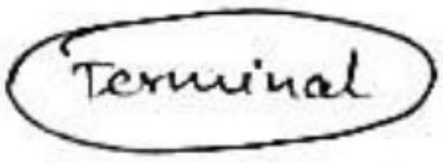
When to use flow-chart :-

* To develop understanding how a process is done.
* To study a process for improvement.
* To document a process.
* When planning a project.

Symbols used in Flowchart designing :

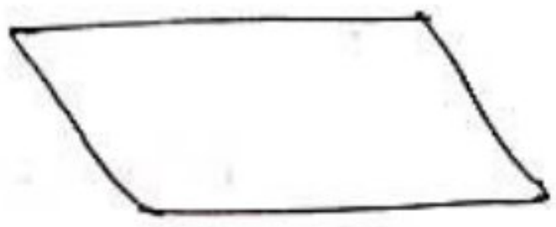
1) Terminal :

The oval symbol is used to indicate start, stop and halt in a program's logic flow. A pause/halt is generally used in a program logic under some error conditions.



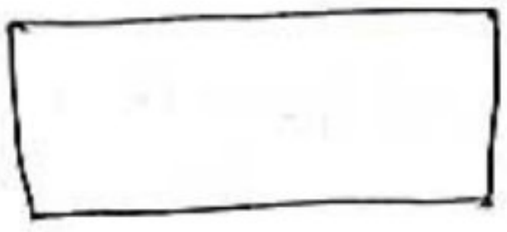
2) Input / Output :

A parallelogram is used to denote the input/output of the program.



3) Processing :

The rectangular box is used to show the processing. ex; arithmatic processes such as adding, subtracting, multiplication and subtraction division are indicated by process symbol.



4) Decesion :

Diamond symbol represents the decision point. It can be either yes/no or true/false type decision.

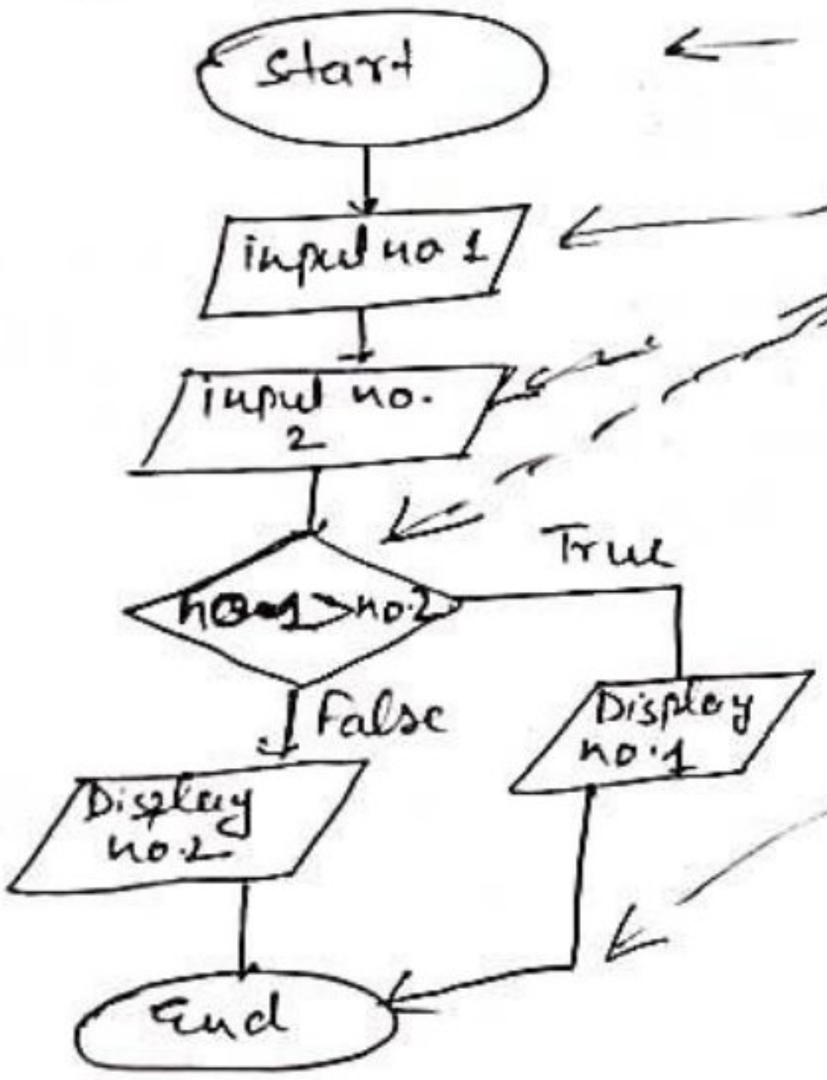
5- **Connectors:**

whenever flow-chart is complex and it spreads over more than 1 page, then we can use connectors to avoid the confusion. It is represented by a circle.

6) **Flow-Lines:**

Flow-Lines indicate the exact sequence in which instructions are executed. Arrow represents the direction of flow of control and relationship among different symbols of flowchart.

Ex:- Draw a flowchart to input two no.s and find largest.



First write algorithm then draw flowchart.

Step 1 → start

step 2 → Read two no's.

step 3 → use comparison operator and check if a is greater or b is greater.

If a is greater display a is greater, otherwise b is greater.

step 4 - stop.

Advantages of flowchart

1- Better way of communicating logic of the system.

2- Acts as a guide for blueprint during program designed.

3- Analysis of programs becomes easy using flowcharts.

4- Provides better documentation.

Disadvantages

1- Difficult to draw flowcharts for large and complex programs.

2- No standard to determine the amount of detail.

3- Difficult to reproduce flowcharts.

4- Difficult to modify the flow-charts.

Pseudo Codes:

* It is a readable description of what a computer program or algorithm must do.
* It is expressed in a formally-styled natural language rather than in a programming language.
* Generally it is written in symbolic code which must be translated into a programming language before it is executed.

Advantages: ~~How to write Pseudocodes~~

* Improves the readability of any algo^{m}.
* It is a bridge b/w the program and the algorithm or flowchart.
* Explains what exactly each line of program should do. and makes code construction phase easier.

How to write a Pseudocode? / Rules for writing the pseudocode.

* Pseudocode doesnot follow a strict or standard way of being written. However there are some standard conventions that need to be followed while writing pseudocodes.

These are as follows —

1- Use capital words for reserved commands or keywords. ex: if we have to write IF--ELSE statement then make sure IF and ELSE must be in capitals.

2- Write only one statement per line.

3- Use indentation for the block body.

4- Be specific while writing a statement, use plain english to provide a particular description.

Constructs of Pseudocode:

1- SEQUENCE - represents linear task sequentially performed one after the other.

WHILE - a loop with a condition at its begining.

REPEAT-UNTIL - a loop with a condition at the bottom.

FOR - another way of looping.

IF-THEN-ELSE - a conditional statement changing the flow of the algo^{m}.

CASE - Generalization form of IF-THEN-ELSE.

Ex: Write a pseudocode to find the largest among two numbers. 2.10

```
BEGIN
    READ n1, n2
    IF n1>n2
            WRITE "n1 is Maximum"
    ELSE
            WRITE "n2 is Maximum"

    ENDIF
```

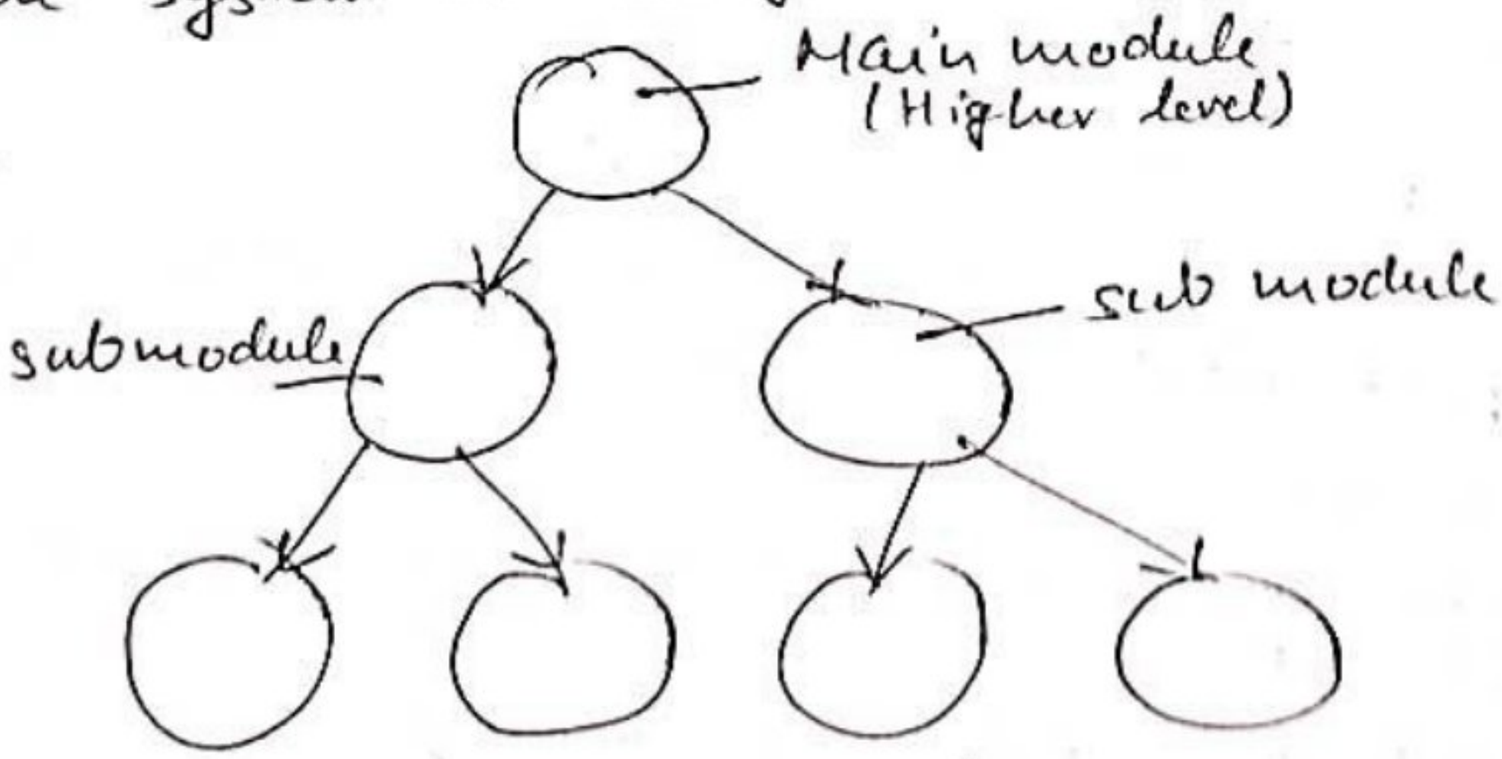
Design strategies :-

* Function Oriented Design.
* Object Oriented Design.
* Top-down and bottom-up design.

Function Oriented Design :

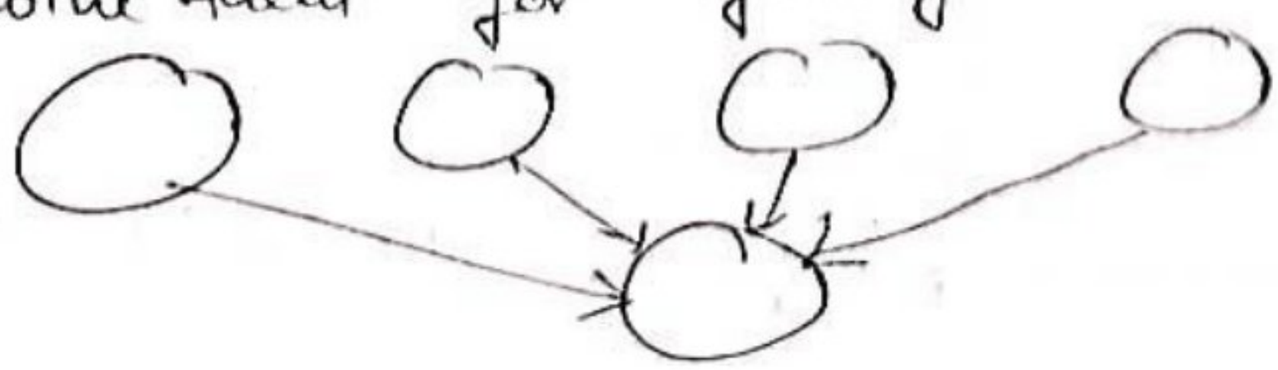
* It is an approach to s/w design in which the main module is decomposed into its submodules where each module has clearly defined functions.
* The system is designed from functional point of view.

Ex:



lower level module.

Now after specifying each module in detail we can combine them for getting the solution.



Function Oriented Design Strategies:-

1) DFD (Data flow Diagram)
2) Structured Charts
3) Pseudo Code

} Already studied in detail previously.

4) Data Dictionaries:-

* Data Dictionaries are simply repositories to store infor about all data items used in DFDs.
* At the requirement stage, data dictionary contains data items. of different types.
* Generally it includes the name of the item, aliases (other name for items), description/purpose, related data items, range of values, data structure definition/form etc.

Advantages of function Oriented Design approch:-

* It is simple and easy.
* Less expensive
* Easier to modify

Object Oriented Design :

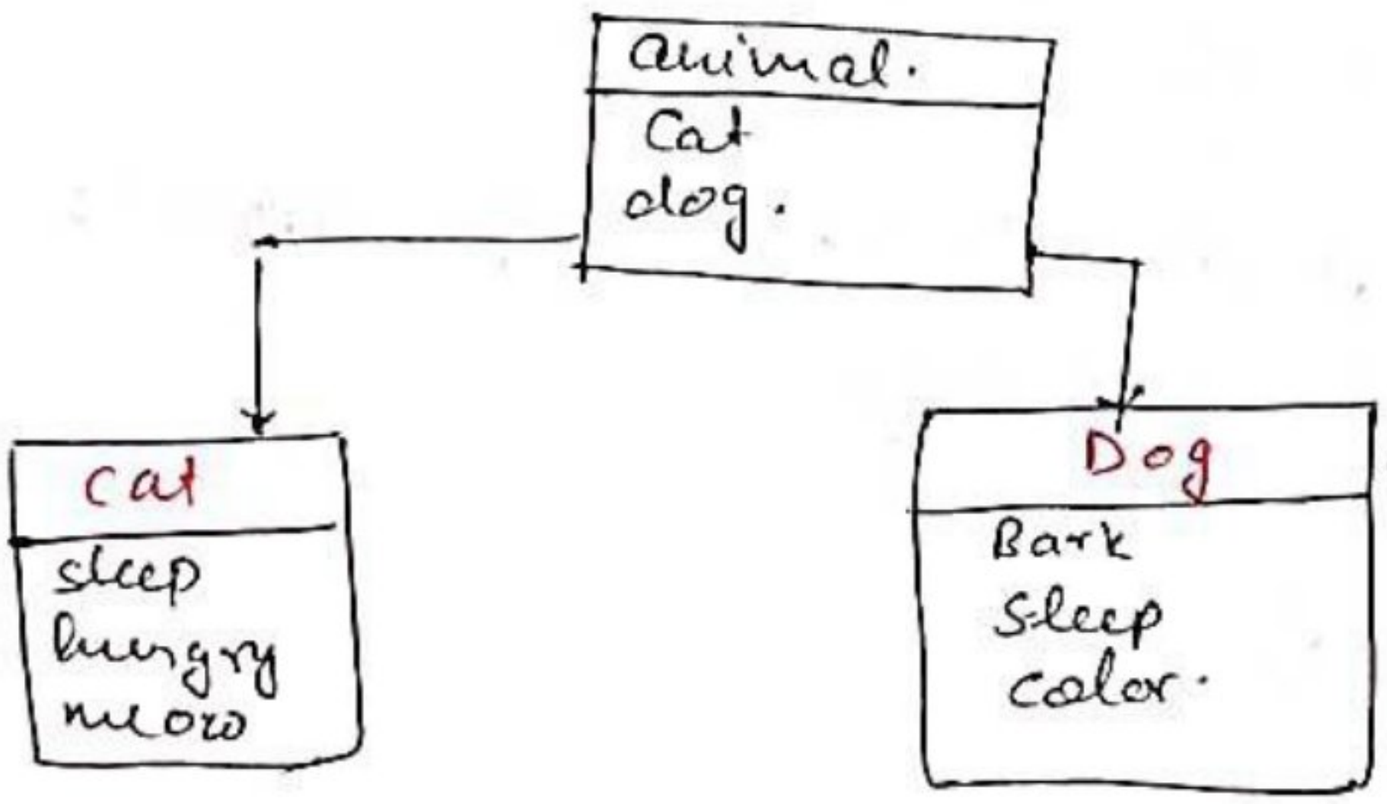
* Object oriented design is the process of using an object-oriented methodology to design a computing system or application.

* In this method, the system is viewed as a collection of objects (i.e. entities). Each object has their own state and the object handles their state data only.

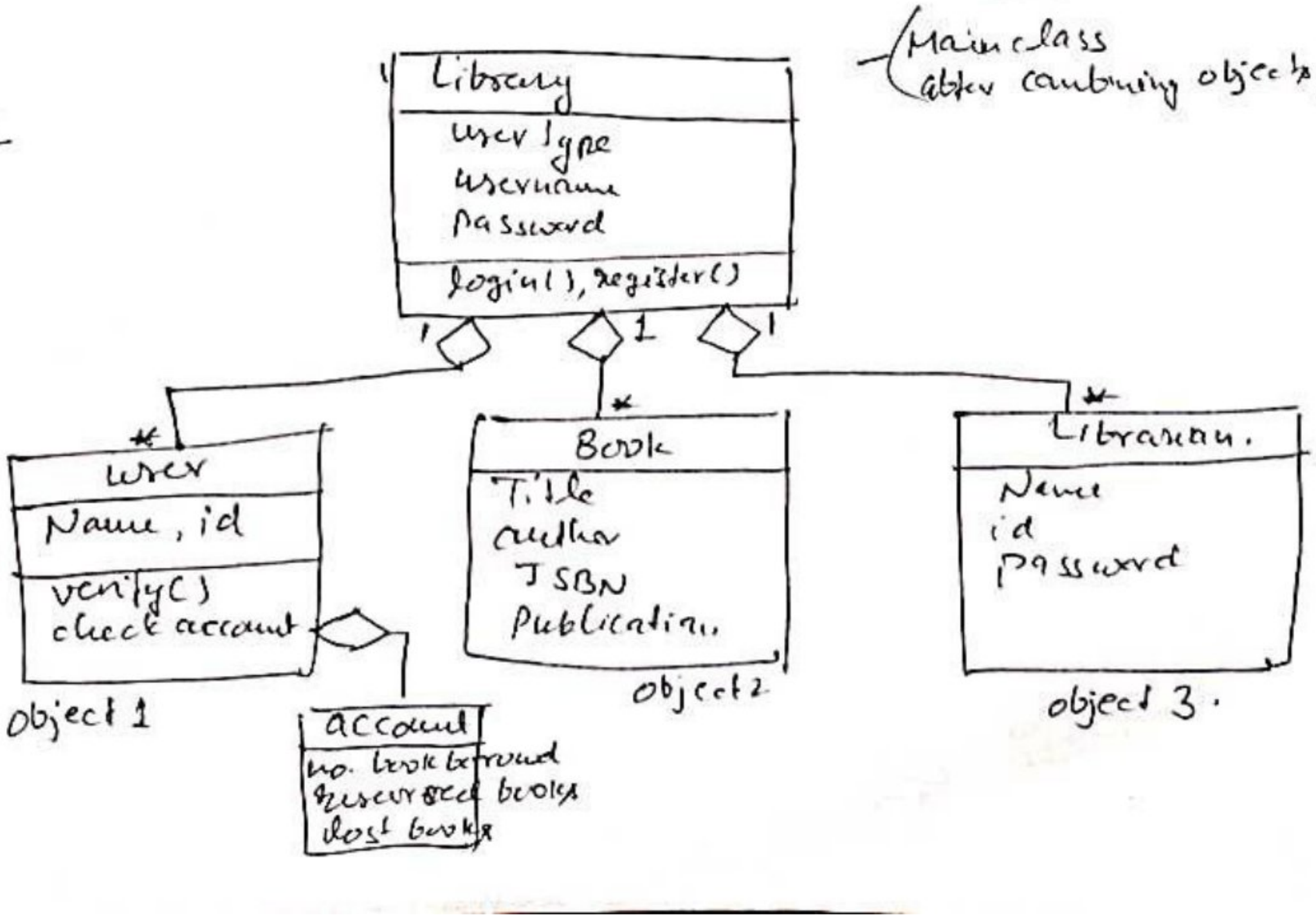
* The tasks defined for one purpose cannot alter/change data of other objects.

* Similar objects are grouped to form classes, or we can say every object is member of some class.

Ex:



Ex:



Concepts used in object-oriented Design:

1. Objects:

* All the entities that are involved in design are k/a objects.
  ex: person, bank, company users etc.
* Every entities are having attributes associated with it and has some methods to perform on the attributes.

2. Classes:

Classes are generalized description of objects.

A class defines all the attributes, which objects are having and methods also, which represents the functionality of the object.

3. Messages:

* Objects can communicated by Message Passing.
* Messages are implemented as procedure or function calls.

4. Abstraction:

An OOD abstraction is used for handling complexity.

It removes irrelevant data and amplifies only essential data.

5 Encapsulation:

* Encapsulation is the concept of hiding info.
* The data and operation are linked into single unit.
* Encapsulation clubs essential info together and also restricts access to the data and methods from outside world.

6) Inheritance:

It is the procedure in which one class inherits the attributes and methods of another class.

It makes it easier to define a specific class and to create generalized classes from specific ones.

7 Polymorphism:

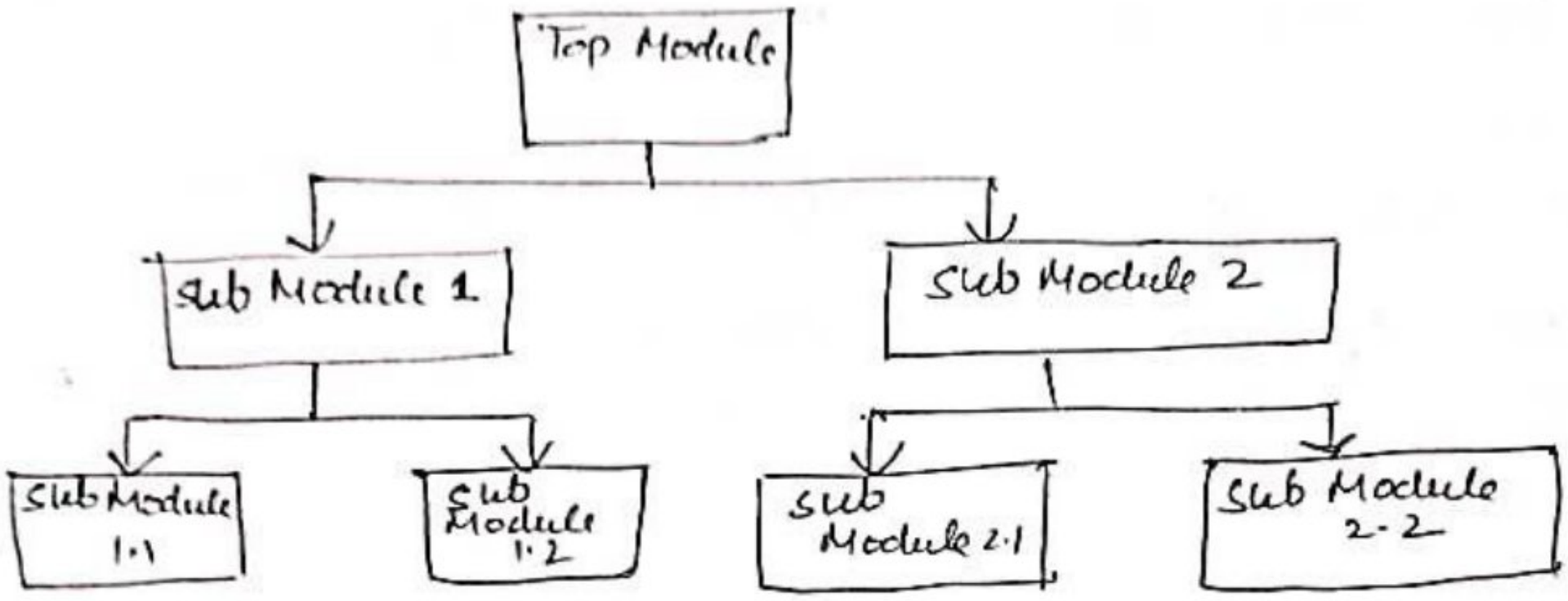
It same entity behaves differently in different scenario.
ex: '+' operator performs addition when used with numbers,

```
int a = 5
int b = 6;
int sum = a + b; // sum = 11
```

```
String firstname = "abc";
String lastname = "xyz";
String name = firstname + lastname;
```

Top down Design Approach / Stepwise design / stepwise refinement

* In this design approach, a system is broken down into its subsystems. Each subsystem is then refined into greater detail until the entire specification is reduced to base elements.
* Testing takes place from top to bottom



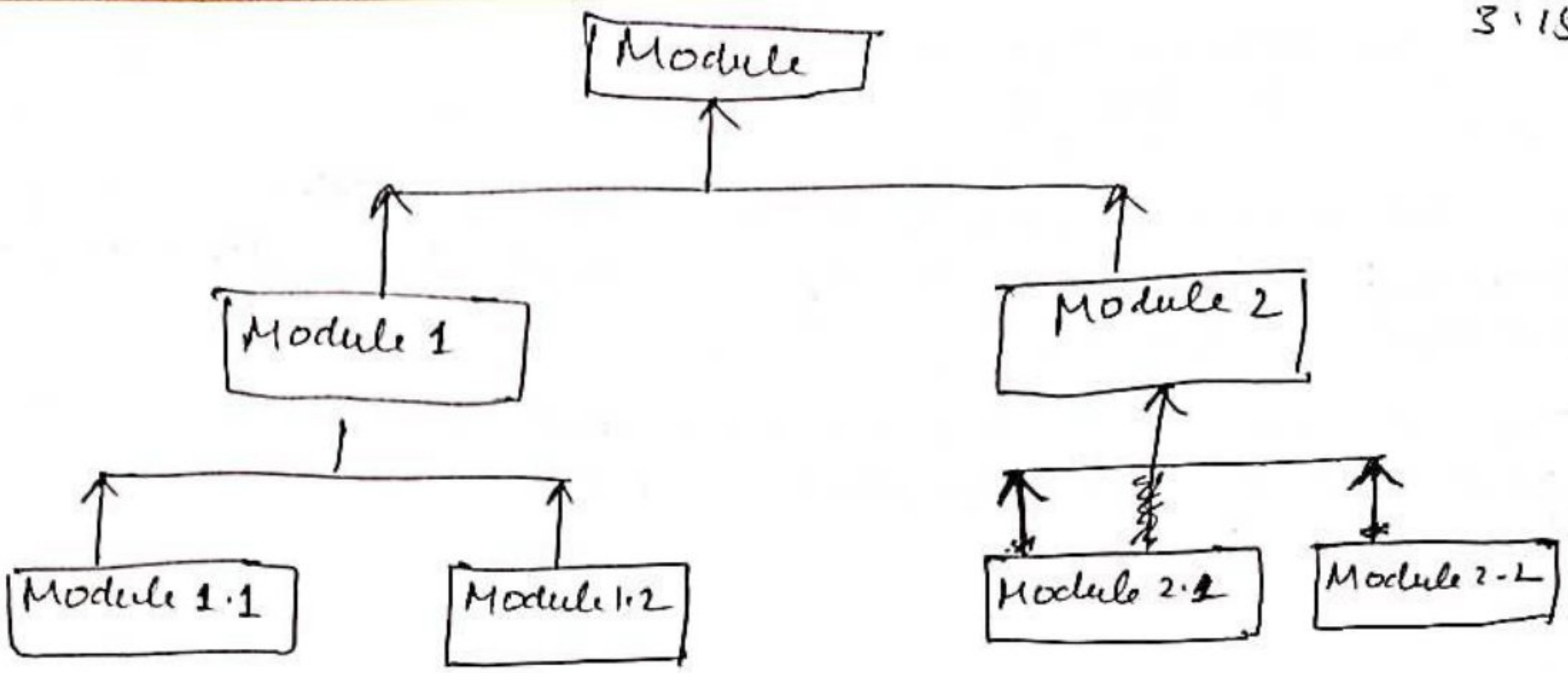
Advantages:

* Break down of Main problem into different parts will help us to identify what needs to be done.
* At each step, new part becomes less complex therefore easy to understand and solve.
* parts can be reused to get the solutions.
* By dividing parts we can allocate them to many persons for solutions.

Bottom UP design Approach:

This approach is defined as piecing together of the submodules inh one to give rise to more complex systems

In this approach the individual base elements of the system are first specified in great detail. The the elements are linked to form the complete system as single unit.



Advantages:

* Increases collaboration and benifits are realized in early phases.
* Flexibility as we have to design individual module separately.
* Fault localization is easy.
* Higher accuracy.
* Interface defects are detected at early stages.
* Observation and maintenance is easy.

S/w Measurements:

S/w Measurement deals with measuring the size, quantity, amount or dimension of a particular attribute of a product or process

The process of S/w measurement is defined and governed by ISO standard, ISO 15939.

Need:

* To ensure that the s/w is bug free before the release.
* To Anticipating future qualities of process or product.
* To Enhance the quality of a product or process.

Classification of Measurement:

1) Direct Measurement:
in this the product, process or thing is measured using direct standards.

2) Indirect Measurements:
in this the quantity or quality to be measured using related ~~prom~~ parameters

S/w Metrics:

It provides the functions by which we can measure the attributes of s/w.

Characteristics:

1- Quantitative: Can be expressed in values.
2- Understandable: easily understood.
3- Applicability: applicable to the phases of s/w development.
4- Repeatable: consistent in nature, no change in values if used repeatedly.
5- Economical:
6- Language independent.

Classification of S/w Metrics:

Product Metrics

* Describes the characteristics of the product.
* Generally measures —
  - Size
  - Complexity
  - Design Features
  - Performance
  - Quality level
  - Reliability
  - Functionality.

Process Metrics

* Used to improve the development process and maintenance activities of S/w.
* Generally include the measurements of —
  - Effort required
  - Time to produce the product.
  - No. of defects found.
  - Tools and tech.
  - Quality
  - Efficiency.

Project Metrics.

Describes project characteristics & execution.

In general it estimates —

- No of S/w developers. staffing patterns
- cost
- schedule
- Productivity.
- Quality.
- Assess status of ongoing project.

Size Oriented Measures/Metrics: (LOC) → Line of Code

* Generally used for measuring the size of the S/w product.
* Any line in code apart from comments and blanks, includes header, declaration as well as executable & non-executable statements.
* For OSCs KLOC (Thousand Line of codes) for counting the measures. i.e LOC is considered to be the normalization value.
* The various parameters that are counted in size oriented metrics are —

  Size = kilo/thousand line of code (KLOC).

  Effort = Person/month.

  Productivity = KLOC/person-month.

  Quality = Number of fault/KLOC

  Cost = $/KLOC.

  Documentation = pages of documents/KLOC.

Que:-

Project	Effort	$/K	KLOC	Pages	error/fault
A	48	139	20	545	39
B	58	142	30	635	53
C	63	165	40	333	22

3.18

Three projects are given with the details of their effort, KLOC, no of pages of project and their no of errors. Calculate the various size oriented metric measures using the details provided into the table.

The various measures can be calculated as —

size = KLOC.
- Size of project A = 20 KLOC or 20,000 LOC
- " " " B = 30 KLOC or 30,000 LOC
- " " " C = 40 KLOC or 40,000 LOC

Effort =
- Effort for A = 48
- " " B = 58
- " " C = 63

productivity =
- for A = KLOC / effort = 20/48 KLOC/person month.
- B = " = 30/58
- C = " = 40/63

Quality =
- for A = No. of faults/KLOC = 39/20
- B = " = 53/30
- C = " = 22/40

Cost =
- for A (cost) = $/KLOC = 139/20
- for B (cost) = $/KLOC = 142/30
- for C (cost) = $/KLOC = 165/40

Documentation =
- A Pages of documentation/KLOC = 545/20
- B = 635/30
- C = 333/40

Halestead's Software science / Halestead Complexity Measures

* GH is a type of size oriented metric (Because depends on counting the s/w line of code (LOC)).
  Introduced by Maurice Howard Halestead in 1977.

* Assumption of this metric is that, a computer program is the collection of instruction and the program is the implementation of an algorithm.

* The algom can be described as the set of operators and operands. So everything will revolve around the counting of operands and the operators in the program.

So for a given problem let —

n1 = the number of distenct operators
n2 = the number of distinct operands.
N1 = the total number of operators.
N2 = the total number of operands.

Now by using these values, various measures can be calculated.

So, the various Halestead metrics are —

1) Halestead Program length (N) = N1+N2
2) " Program Vocabulary (n) = n1+n2
3) Estimated Program length (N) = $n_1 \log_2 n_1 + n_2 \log_2 n_2$
4) Program Volume (V) = $N \times \log_2 n$ (space necessary for storing the program.
5) Program Difficulty (D) = $\frac{n_1}{2} \times \frac{N_2}{n_2}$ (how difficult to handle the program).
6) Program Effort (E) = $D \times V$

(It describes the amount of mental activity needed to translate the existing algom into implementation in the specified programming language.)

7) Intelligence Content (I) = $\frac{V}{D}$ (amount of intelligence presented in program) measure the complexity independent upon the language used.

Programming Time (T) = $\frac{E}{S}$ in minutes

$$T = \frac{E}{18 \times 60} \text{ minutes.}$$

Here T denotes the time required to translate the algorithm into implementation i.e. in programming language.

$(5 <= S <= 20)$ 3:20

Halestead chooses S = 18 and S is measured in seconds.

S = stroud number. (processing rate of human brain)

Example:

```
main()
{ int a, b, c, avg;
  scanf("%d%d%d", &a, &b, &c);
  avg = (a+b+c)/3;
  printf("avg = %d", avg);
}
```

Soln → For this program calculate the operators and operands.

operators	occurences	operands	occurences
main()	1	a	3
()	3	b	3
{}	1	c	3
int	1	avg	1
scanf	1	"%d%d%d"	1
&	3	3	1
=	2	"avg = %d"	1
+	2		
/	1		
printf	1		
,	7		
;	4		

$n_1 = 12$ $N_1 = 27$ $n_2 = 7$ $N_2 = 15$

Program length(N) = N1 + N2 = 27 + 15 = 42

Program vocabulary(n) = n1 + n2 = 12 + 7 = 19

Estimated Program length $\hat{N} = n_1 \log_2 n_1 + n_2 \log_2 n_2 = 62.7$

Volume(V) = $N \times \log_2 n$

$= 42 \log_2 19 = 178.4$

Difficulty(D) = $\frac{n_1}{2} \times \frac{N_2}{n_2}$

$= \frac{12}{2} \times \frac{15}{7} = 12.85$

Note: As it is not possible to directly calculate the value of $\log_2 n$, so convert it to $\log_{10}$ by dividing it 0.3010

$\log_2 n = \frac{\log_{10} n}{0.3010}$

Effort(E) = $12.85 \times 178.4 = 2292.44$

Time required(T) = $\frac{2292.44}{18} = 127.357$ second

$= \frac{127.357}{60}$ minutes.

No. of delivered bugs (B) = $\frac{E^{2/3}}{3000} = 0.05$

or $B = \frac{V}{3000}$ is accepted.

Ex.2

```
int sort (int x[], int n)
{
  int i, j, save, im1;
  /* This function sorts array in ascending order. */
  if(n<2) return 1;
  for(i=2; i<n; i++)
  {
    im1 = i-1;
    for(j=1; j<im1; j++)
      if(x[j] < x[i])
      {
        save = x[j];
        x[j] = x[i];
        x[i] = save;
      }
  }
  return 0;
}
```

Solⁿ:- Count the no. of operators and operands.

operators	occurences	operands	occurences.
int	4	sort	1
( )	5	x	7
,	4	n	3
[]	7	i	8
if	2	j	7
<	2	save	3
;	11	img	3
for	2	2	2
=	6	3	3
-	1	1	1
<=	2	0	
++	2		
return	2		
{ }	3		
	$n_1 = 14$		$n_2 = 10$
	$N_1 = 53$		$N_2 = 38$

So $N = N_1 + N_2 = 91$
$n = n_1 + n_2 = 24$

Rest of the measure you can calculate by using all the formulas. by simply putting the values of n_1, N_1, n_2, N_2.

Function Oriented Metrics:- Function Point Analysis

Function point analysis is the process of sizing s/w based on the no. of business functions that an applicn must accomplish.

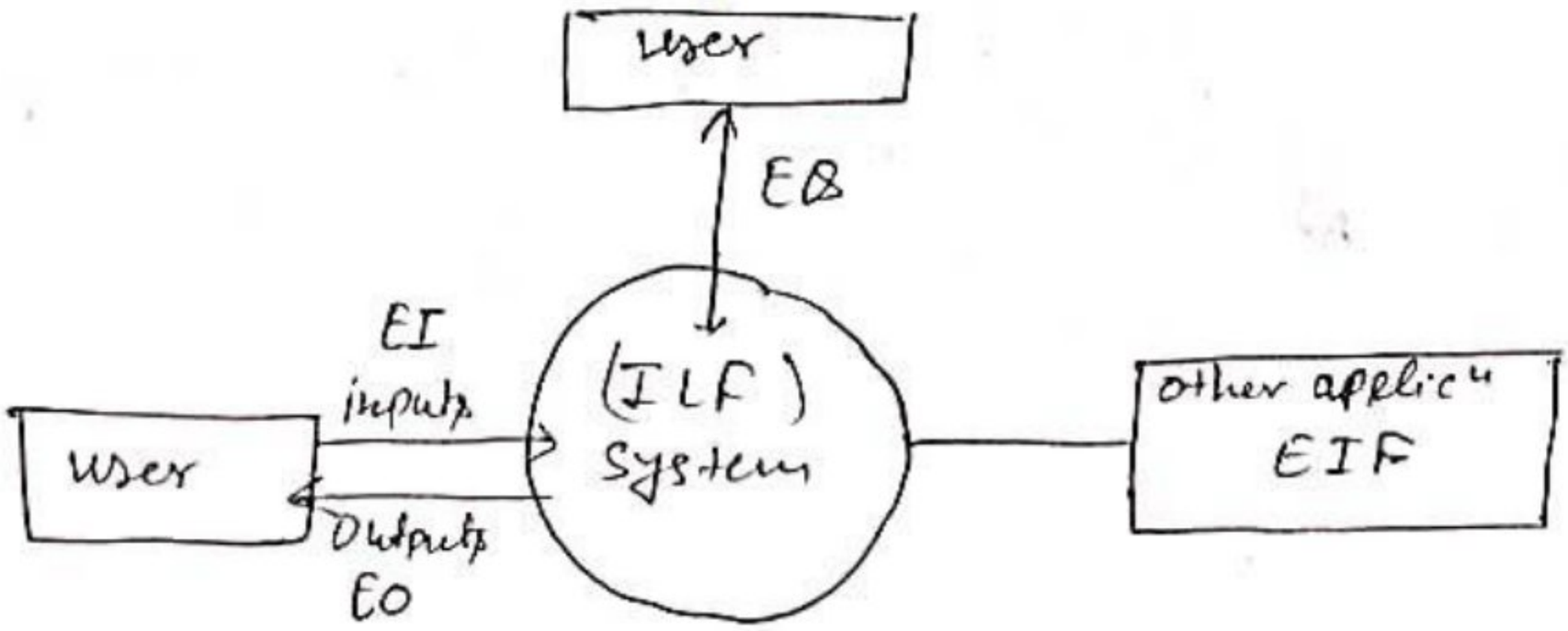
It eliminates the needs of going through each line of code to determine if an applicn/s/w meets requirements or not.

Procedure of FPA (Function Point Analysis):

1) Count the no of and types of the functions used in the application. Various functions used in an application can be put under 5 categories.

(FP attributes)

Measurement Parameters	Examples
1- No. of External Inputs (EI)	Input screen & tables.
2- No. of external Outputs (EO)	O/P screen and reports.
3- No. of external Inquiries (EQ)	prompt and interrupts.
4- No. of internal Files (ILF)	Databases and directories.
5- No. of external interfaces (EIF)	Shared databases & shared routines



All the parameters mentioned above are assigned some weights that have been experimentally determined.