GANPAT UNIVERSITY

U. V. PATEL COLLEGE OF ENGINEERING

2CEIT502 SOFTWARE ENGINEERING

UNIT 9

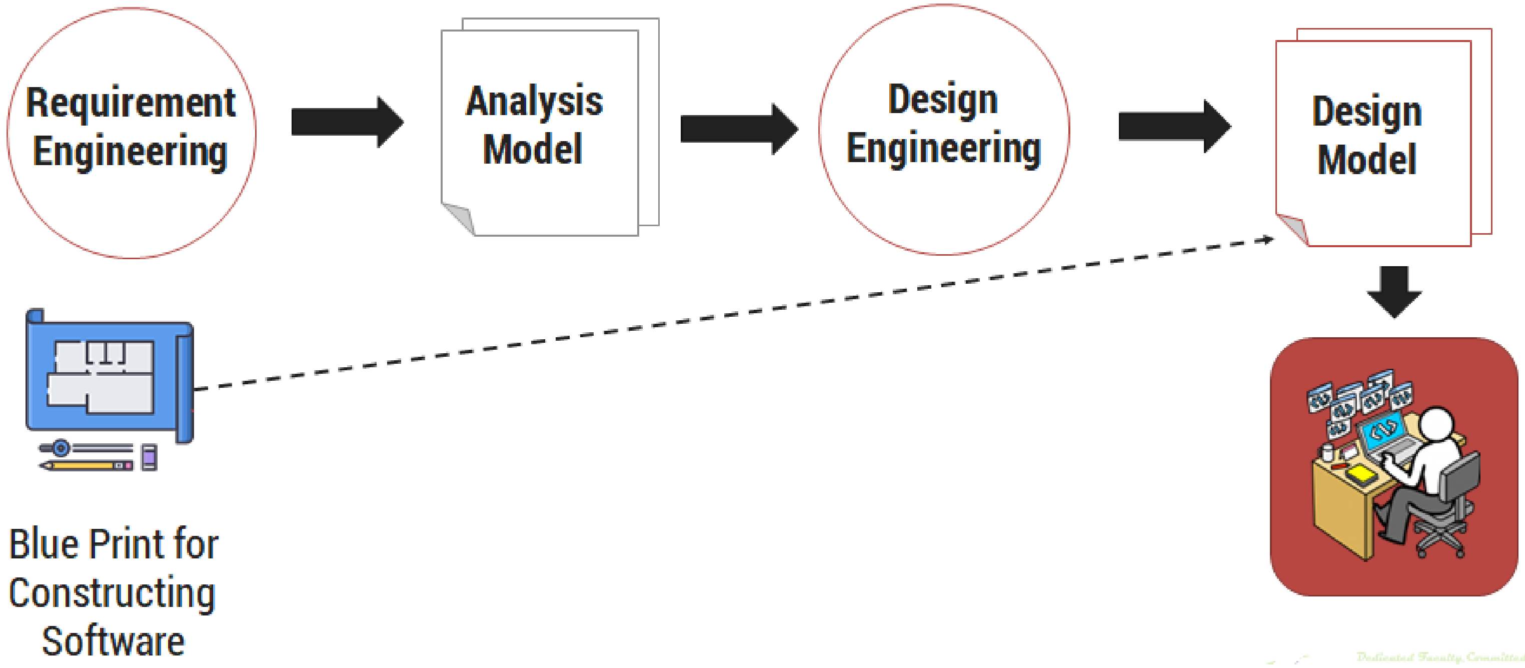
FUNCTION ORIENTED SOFTWARE DESIGN

Prepared by: Prof. Y. J. Prajapati (Asst. Prof in IT Dept. , UVPCE)

Outline

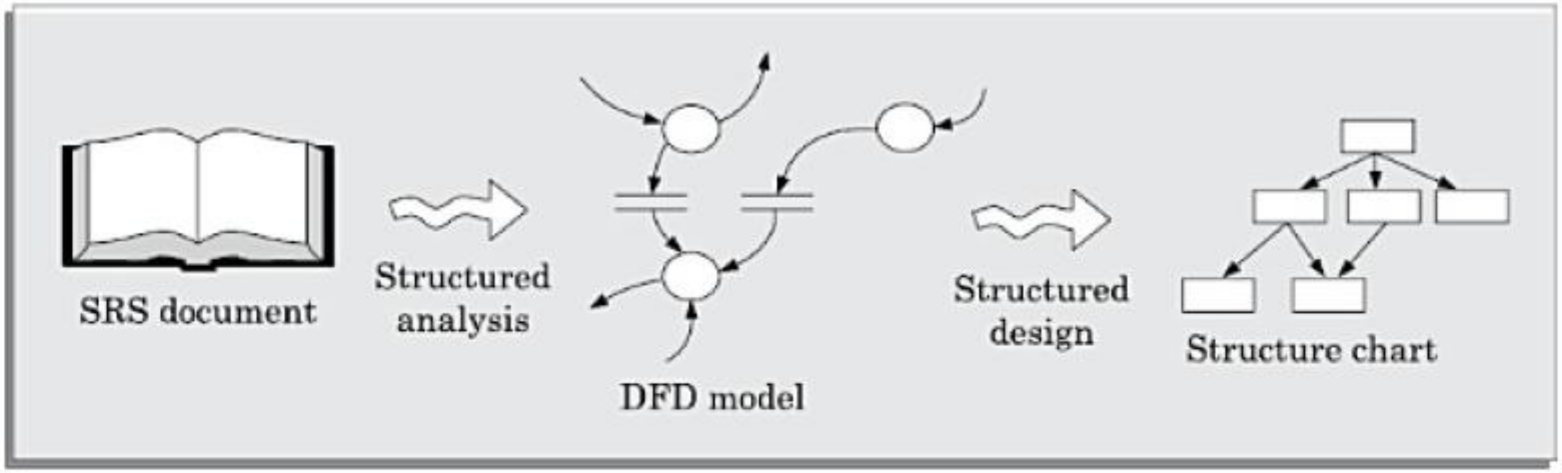
- Overview of SA/SD methodology Structured analysis
- Data flow diagrams (DFDs)
- Structure design

Software Design Process ?



Overview of SA/SD methodology

- As the name itself implies, SA/SD methodology involves carrying out two distinct activities:
  - Structured analysis (SA)
  - Structured design (SD)



Overview of SA/SD methodology

STRUCTURED ANALYSIS

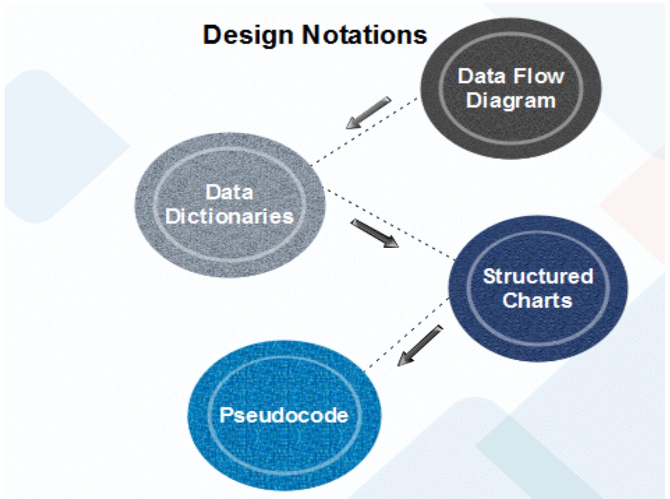
- We have already mentioned that during structured analysis, the major processing tasks (high-level functions) of the system are analysed, and the data flow among these processing tasks are represented graphically.
- Significant contributions to the development of the structured analysis techniques have been made by Gane and Sarson [1979], and DeMarco and Yourdon [1978].

The structured analysis technique is based on the following underlying principles:

- Top-down decomposition approach.
- Application of divide and conquer principle. Through this each high level function is independently decomposed into detailed functions.
- Graphical representation of the analysis results using data flow diagrams (DFDs)

Function Oriented Design

- Function Oriented design is a method to software design where the model is decomposed into a set of interacting units or modules where each unit or module has a clearly defined function. Thus, the system is designed from a functional viewpoint.



Data Flow Diagram

- Data-flow design is concerned with designing a series of functional transformations that convert system inputs into the required outputs. The design is described as data-flow diagrams. These diagrams show how data flows through a system and how the output is derived from the input through a series of functional transformations.

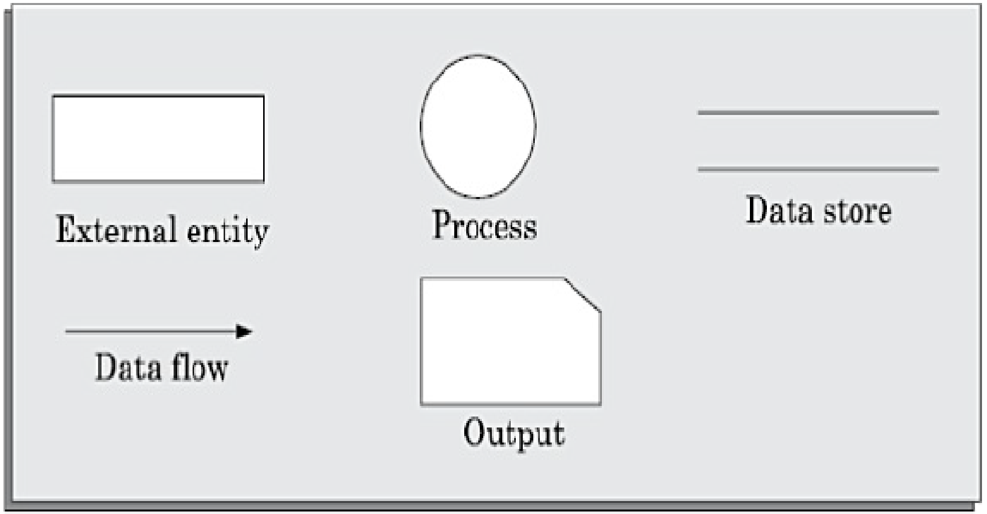
Data Flow Diagrams (DFDs)

- The DFD (also known as the bubble chart) is a simple graphical formalism that can be used to represent a system in terms of the input data to the system, various processing carried out on those data, and the output data generated by the system.

- A data flow diagram (DFD) maps out the flow of information for any process or system. It uses defined symbols like rectangles, circles and arrows, plus short text labels, to show data inputs, outputs, storage points and the routes between each destination.

Data Flow Diagrams (DFDs)

- Primitive symbols used for constructing DFDs



External entity

- The external entities are essentially those physical entities external to the software system which interact with the system by inputting data to the system or by consuming the data produced by the system.
- An external entity such as a librarian, a library member, etc. is represented by a rectangle.
- In addition to the human users, the external entity symbols can be used to represent external hardware and software such as another application software that would interact with the software being modelled.

Data Flow Diagrams (DFDs)

Data flow symbol

- A directed arc (or an arrow) is used as a data flow symbol
- A data flow symbol represents the data flow occurring between two processes or between an external entity and a process in the direction of the data flow arrow.
- Data flow symbols are usually annotated with the corresponding data names.

Data store symbol

- A data store is represented using two parallel lines.
- It represents a logical file.
- A data store symbol can represent either a data structure or a physical file on disk.
- Each data store is connected to a process by means of a data flow symbol.
- The direction of the data flow arrow shows whether data is being read from or written into a data store.
- An arrow flowing in or out of a data store implicitly represents the entire data of the data store and hence arrows connecting to a data store need not be annotated with the name of the corresponding data items.

Output symbol

- The output symbol is used when a hard copy is produced.

Data Flow Diagrams (DFDs)

Levels in Data Flow Diagrams (DFD)

- The DFD may be used to perform a system or software at any level of abstraction.
- DFDs may be partitioned into levels that represent increasing information flow and functional detail.
- Levels in DFD are numbered 0, 1, 2 or beyond.
- Here, we will see primarily three levels in the data flow diagram, which are: 0-level DFD, 1-level DFD, and 2-level DFD.

- **0-level DFD**
- It is also known as fundamental system model, or context diagram represents the entire software requirement as a single bubble with input and output data denoted by incoming and outgoing arrows.

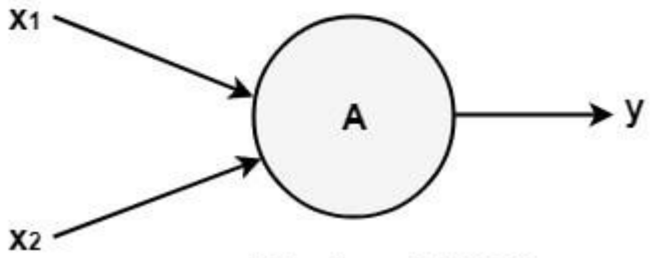


Fig: Level-0 DFD.

0-level DFD

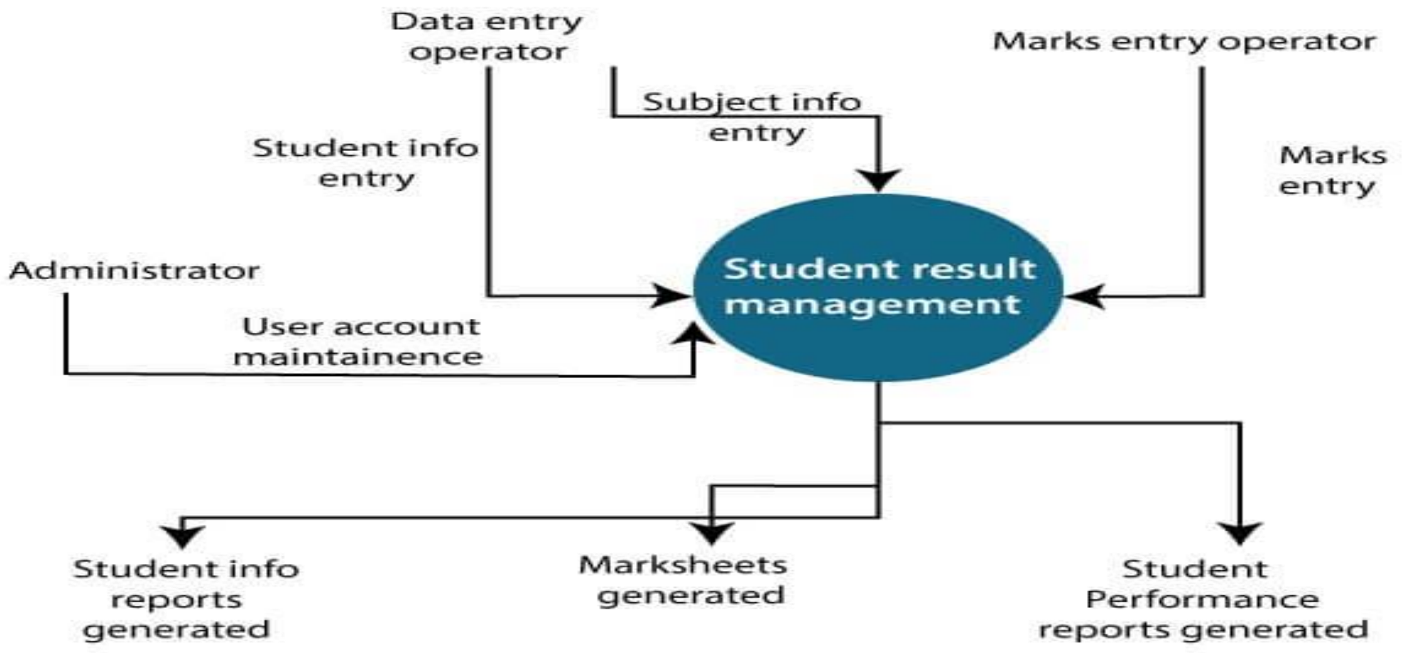


Fig: Level-0 DFD of result management system

1-level DFD

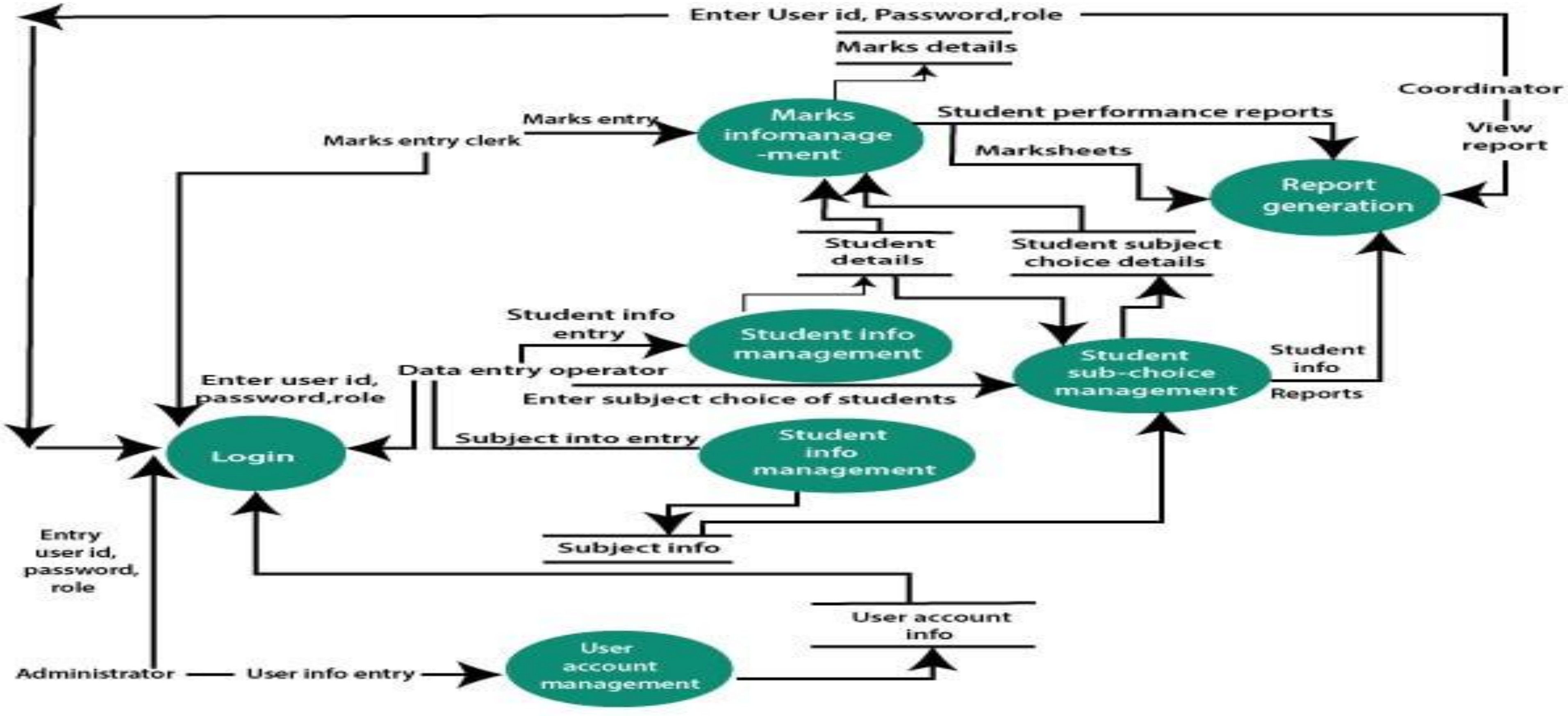
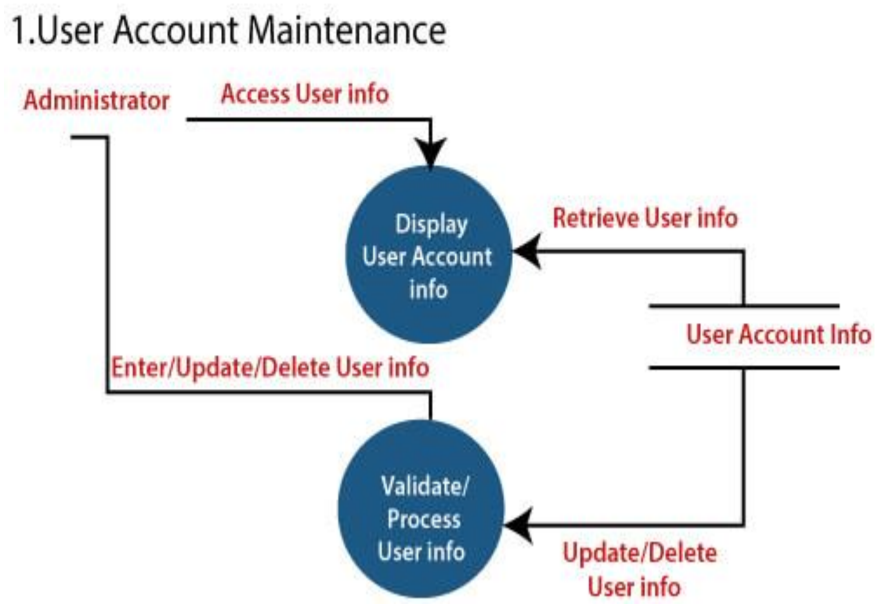
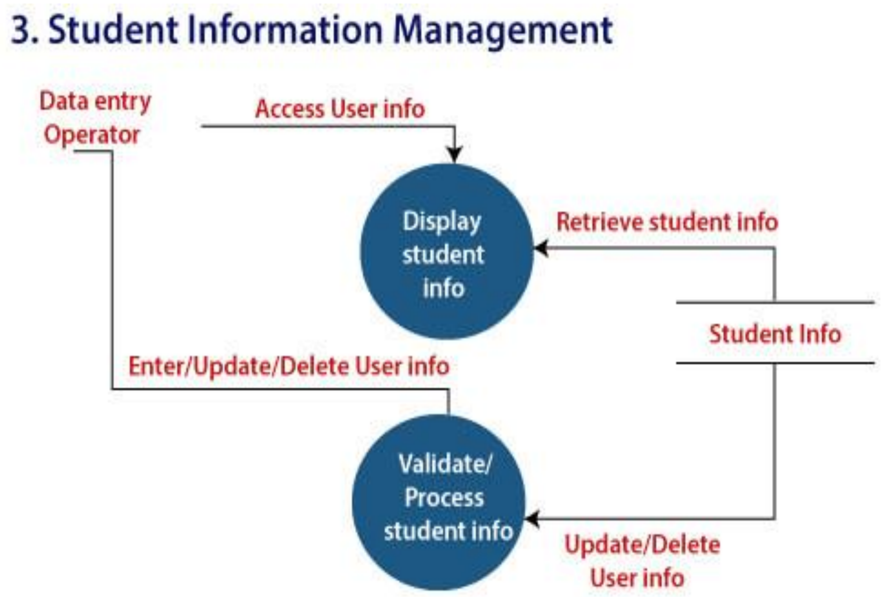


Fig: Level-1 DFD of result management system

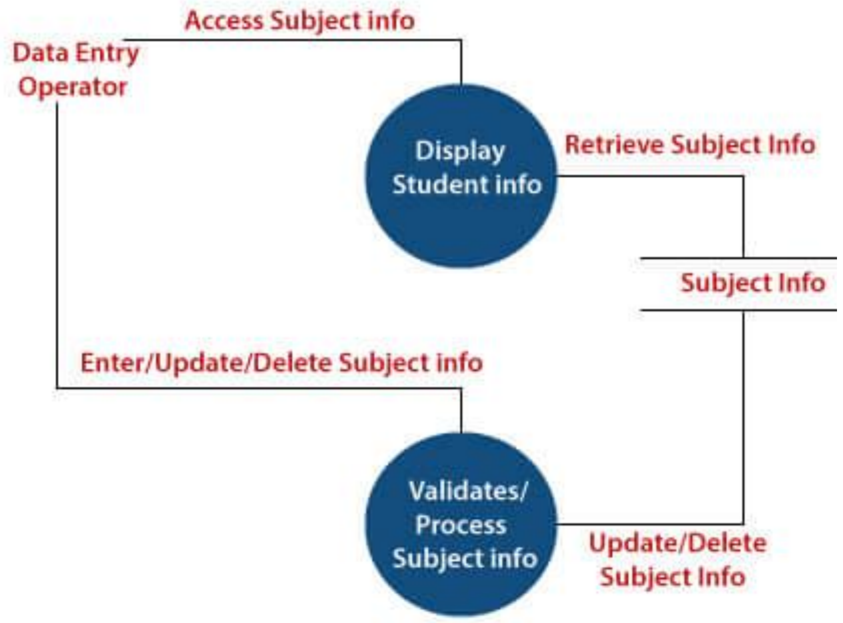
2-Level DFD


1.User Account Maintenance
Administrator
Access User info
Display User Account info
Retrieve User info
User Account Info
Enter/Update/Delete User info
Validate/ Process User info
Update/Delete User info


3. Student Information Management
Data entry Operator
Access User info
Display student info
Retrieve student info
Student Info
Enter/Update/Delete User info
Validate/ Process student info
Update/Delete User info

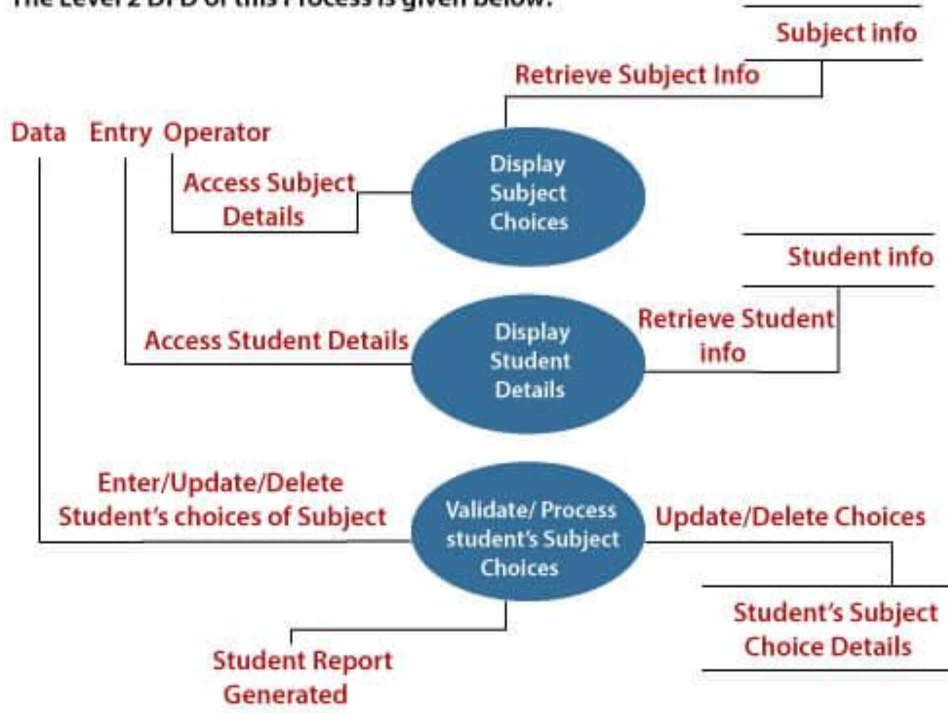
4. Subject Information Management

The level 2 DFD of this process is given below:



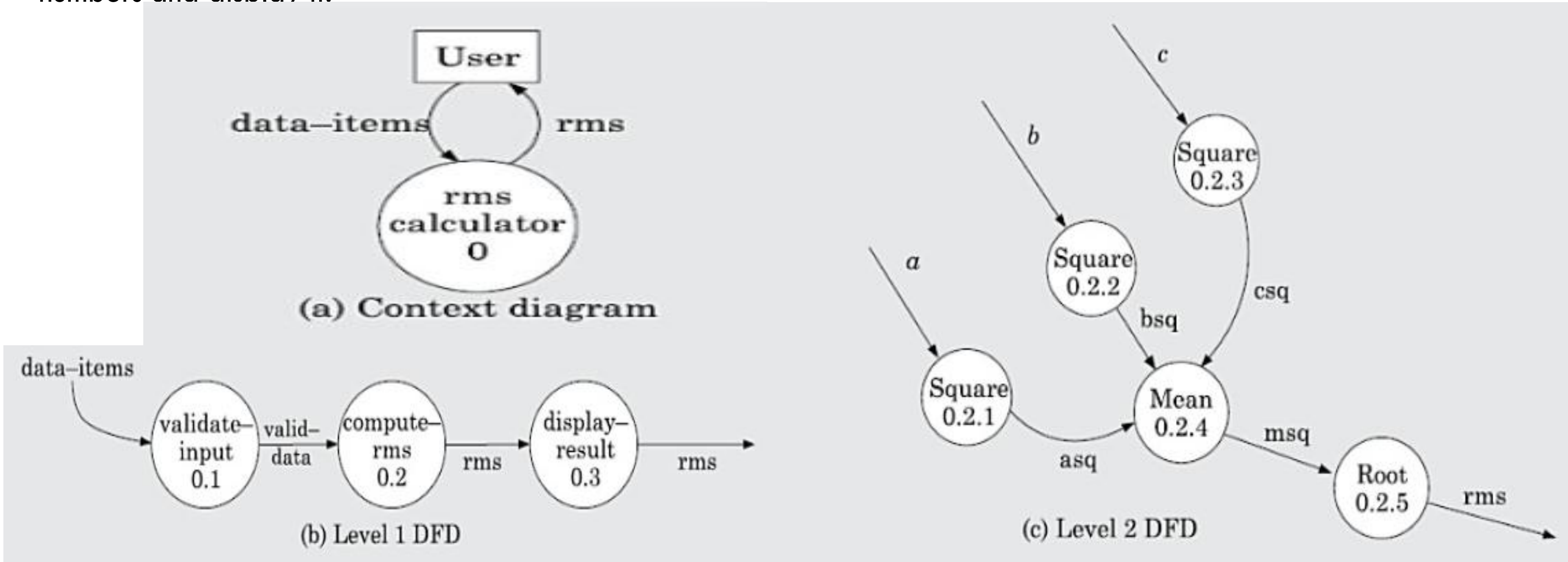
5. Student's Subject Choice Management

The Level 2 DFD of this Process is given below:



Example

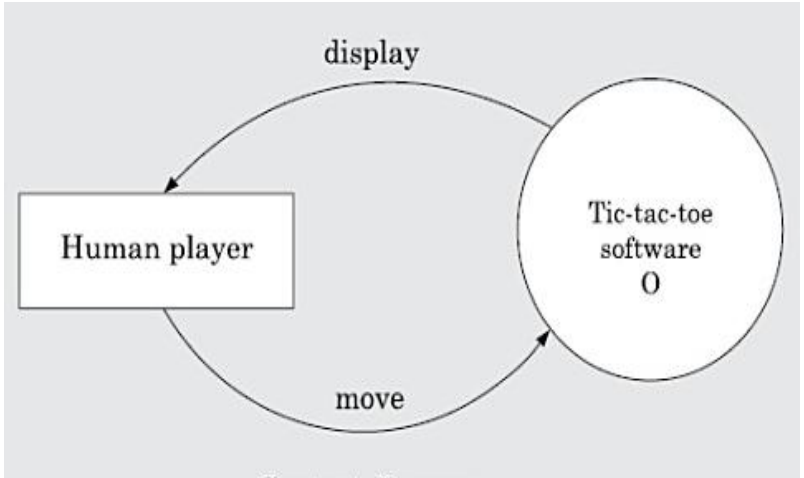
- **(RMS Calculating Software) A software system called RMS** calculating software would read three integral numbers from the user in the range of –1000 and +1000 and would determine the root mean square (RMS) of the three input numbers and display it.



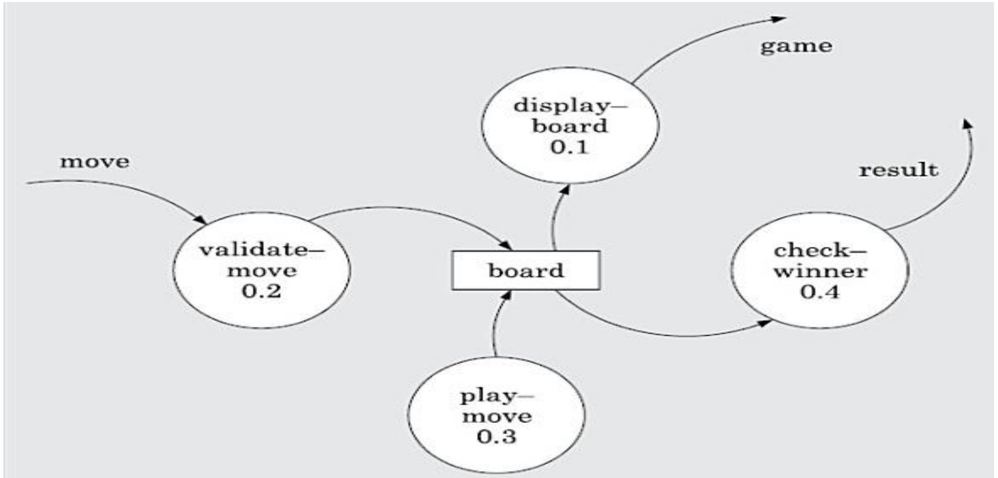
Example

(Tic-Tac-Toe Computer Game)

- Tic-tac-toe is a computer game in which a human player and the computer make alternate moves on a 3 × 3 square. A move consists of marking a previously unmarked square. The player who is first to place three consecutive marks along a straight line (i.e., along a row, column, or diagonal) on the square wins. As soon as either of the human player or the computer wins, a message congratulating the winner should be displayed. If neither player manages to get three consecutive marks along a straight line, and all the squares on the board are filled up, then the game is drawn. The computer always tries to win a game.



Context diagram



Level 1 DFD

Function Oriented Design

Data Dictionaries

- Data dictionaries are simply repositories to store information about all data items defined in DFDs. At the requirement stage, data dictionaries contains data items.
- Data dictionaries include Name of the item, Aliases (Other names for items), Description / purpose, Related data items, Range of values, Data structure definition / form.

- A data dictionary plays a significant role in any software development process because of the following reasons:
- A Data dictionary provides a standard language for all relevant information for use by engineers working in a project. A consistent vocabulary for data items is essential since, in large projects, different engineers of the project tend to use different terms to refer to the same data, which unnecessarily causes confusion.
- The data dictionary provides the analyst with a means to determine the definition of various data structures in terms of their component elements.

Data Dictionaries

Data definition

- Composite data items can be defined in terms of primitive data items using the following data definition operators.
- +: denotes composition of two data items, e.g. a+b represents data a and b.
- [,,]: represents selection, i.e. any one of the data items listed inside the square bracket can occur For example, [a,b] represents either a occurs or b occurs.
- (): the contents inside the bracket represent optional data which may or may not appear. a+(b) represents either a or a+b occurs.
- {}: represents iterative data definition, e.g. {name}5 represents five name data. {name}* represents zero or more instances of name data.
- =: represents equivalence, e.g. a=b+c means that a is a composite data item comprising of both b and c.

Example

RMS Calculating Software

data-items: {*integer*}*3*

rms: float

valid-data:data-items

a: integer

b: integer

c: integer

asq: integer

bsq: integer

csq: integer

msq: integer

Tic-Tac-Toe Computer Game

move: integer /* number between 1 to 9 */

display: game+result

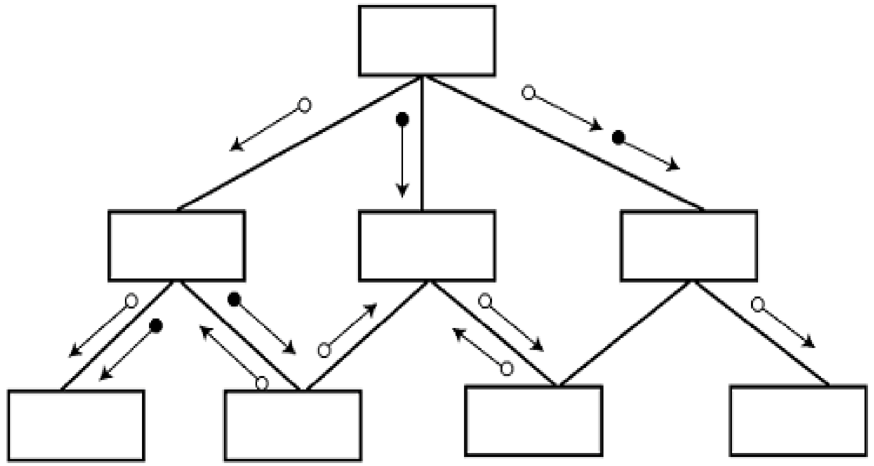
game: board

board: {*integer*}*9*

result: [“computer won”, “human won”, “drawn”]

Function Oriented Design

- **Structured Charts**
- It is the hierarchical representation of system which partitions the system into black boxes (functionality is known to users but inner details are unknown). Components are read from top to bottom and left to right. When a module calls another, it views the called module as black box, passing required parameters and receiving results.
- Structured Chart is a graphical representation which shows:
  - System partitions into modules
  - Hierarchy of component modules
  - The relation between processing modules
  - Interaction between modules
  - Information passed between modules



Hierarchical format of a structure chart

Symbols used in construction of structured chart

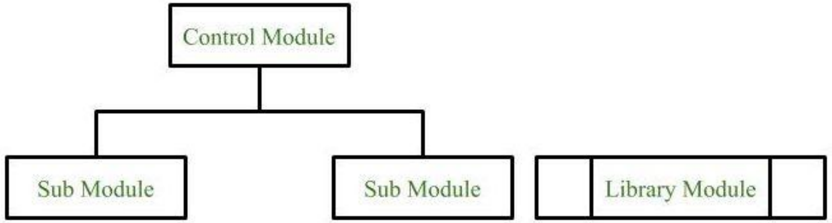
Module

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.

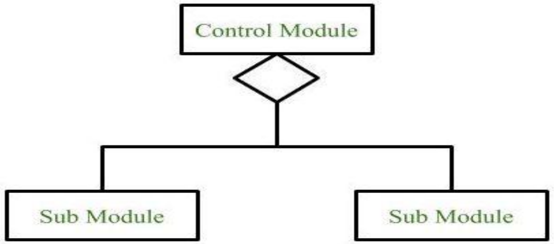
Sub Module
Sub Module is a module which is the part (Child) of another module.

Library Module
Library Module are reusable and invokable from any module.



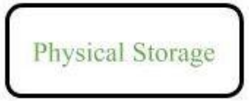
Conditional Call/ Selection

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



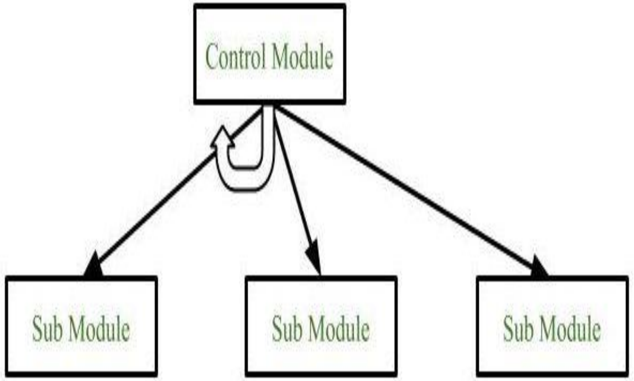
Physical Storage

Physical Storage is that where all the information are to be stored.



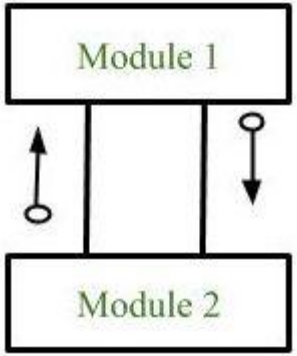
Loop (Repetitive call of module)

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



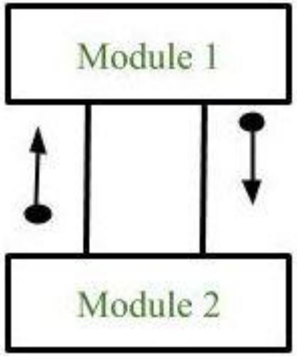
Data Flow

It represents the flow of data between the modules. It is represented by directed arrow with empty circle at the end.

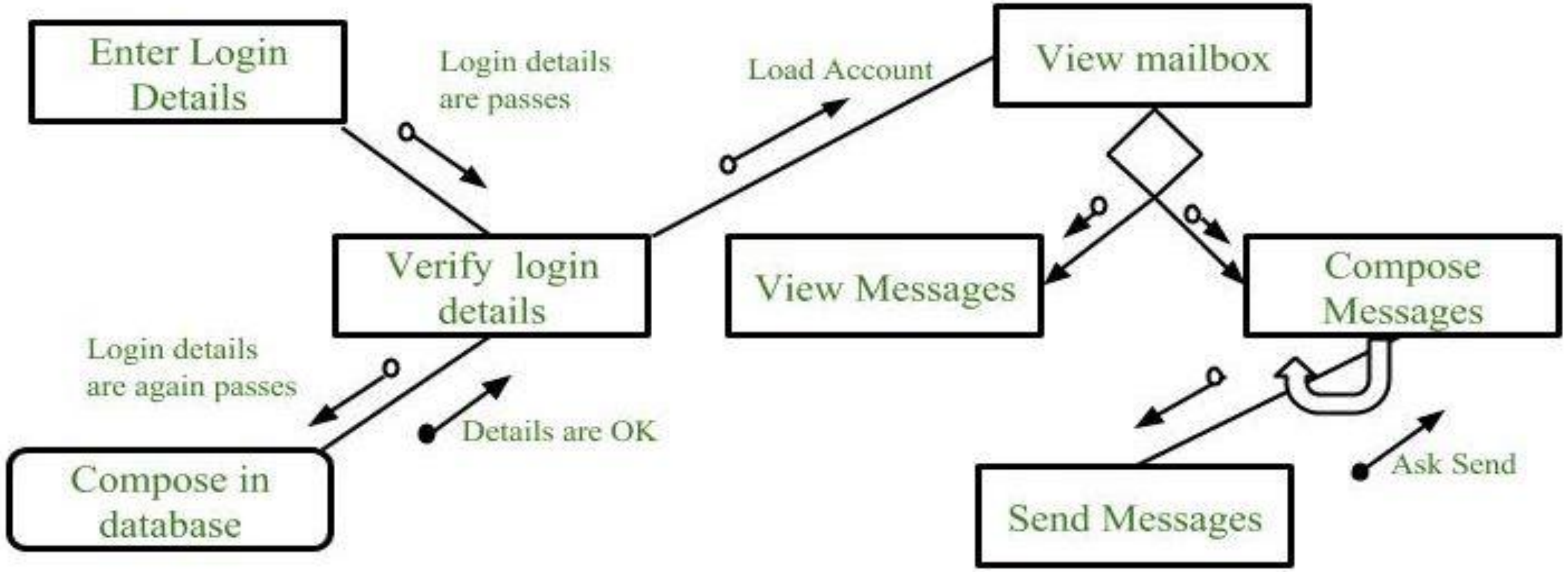


Control Flow

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

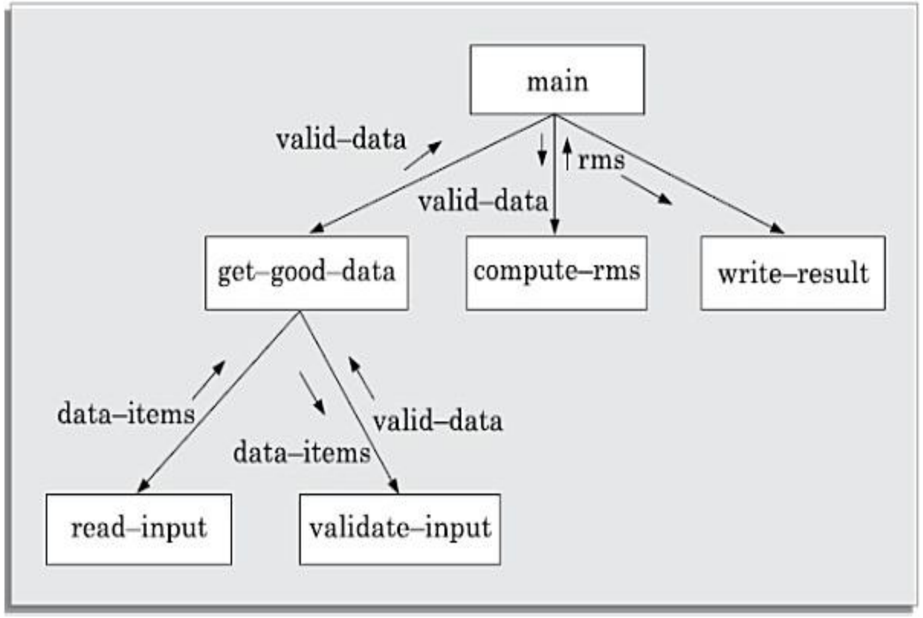


Example : Structure chart for an Email server

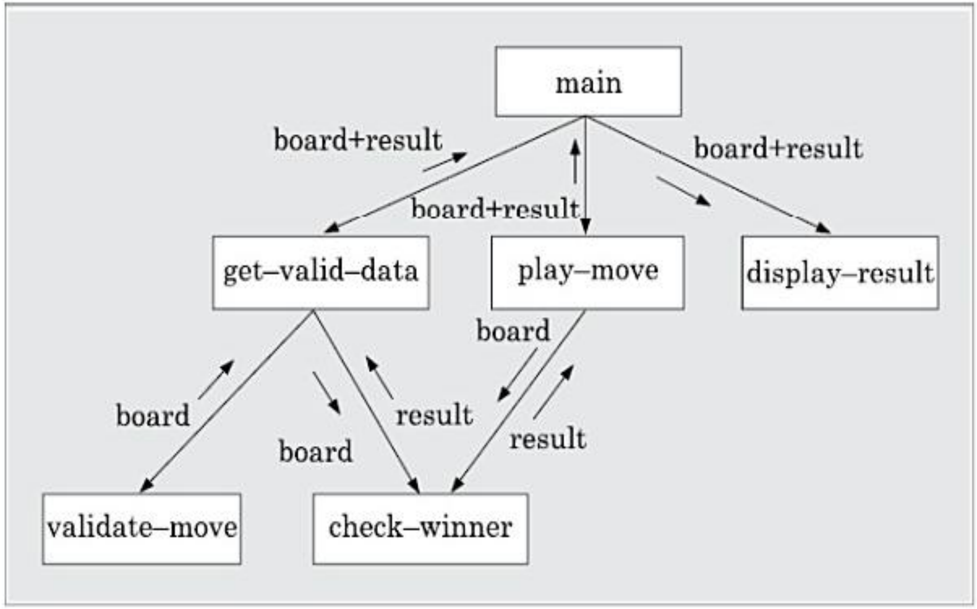


Example

Draw the structure chart for the RMS software



Draw the structure chart for the tic-tac-toe software



Function Oriented Design

- **Pseudo-code**
- Pseudo-code notations can be used in both the preliminary and detailed design phases. Using pseudo-code, the designer describes system characteristics using short, concise, English Language phases that are structured by keywords such as If-Then-Else, While-Do, and End.
- It use keyword and indentation. Pseudo codes are used as replacement for flow charts

THANK YOU!
ANY QUESTIONS?