UNIT – III

Function Oriented Software Design & User Interface Design

Function Oriented Software Design

There are two fundamentally different approaches to software design:

- function-oriented approach
- object-oriented approach

- Function-oriented design techniques are very popular:
  - currently in use in many software development organizations.
- Function-oriented design techniques:
  - start with the functional requirements specified in the SRS document.
- During the design process:
  - high-level functions are successively decomposed into more detailed functions.
  - Technically known as top-down decomposition.
  - The detailed functions are mapped to a module structure.

SA/SD (Structured Analysis/Structured Design)

- SA/SD methodology:
  - has essential features of several important function-oriented design methodologies ---
    - if you need to use any specific design methodology later on,
    - you can do so easily with small additional effort.
- SA/SD technique can be used to perform high-level design.
- **SA/SD methodology consists of two distinct activities:**
  - Structured Analysis (SA)
  - Structured Design (SD)
- During structured analysis:
  - functional decomposition takes place.
- During structured design:
  - module structure is formalized.

Functional decomposition

- Each function is analyzed:
  - hierarchically decomposed into more detailed functions.
  - simultaneous decomposition of high-level data into more detailed data.

Structured analysis

- Transforms a textual problem description into a graphic model.
  - done using data flow diagrams (DFDs).
  - DFDs graphically represent the results of structured analysis.

Structured design

- All the functions represented in the DFD mapped to a module structure.
- The module structure is called as the software architecture.
- Software architecture:
  - refined through detailed design.
  - Detailed design can be directly implemented:
    - using a conventional programming language.

Structured Analysis vs. Structured Design

- Purpose of structured analysis:
  - capture the detailed structure of the system as the user views it.
- Purpose of structured design:
  - arrive at a form that is suitable for implementation in some programming language.

- The results of structured analysis can be easily understood even by ordinary customers:
  - does not require computer knowledge
  - directly represents customer's perception of the problem
  - uses customer's terminology for naming different functions and data.
- The results of structured analysis can be reviewed by customers:
  - to check whether it captures all their requirements.

Structured Analysis

- Based on principles of:
  - Top-down decomposition approach.
  - Divide and conquer principle:
    - each function is considered individually (i.e. isolated from other functions)
    - decompose functions totally disregarding what happens in other functions.
  - Graphical representation of results using
    - data flow diagrams (or bubble charts).

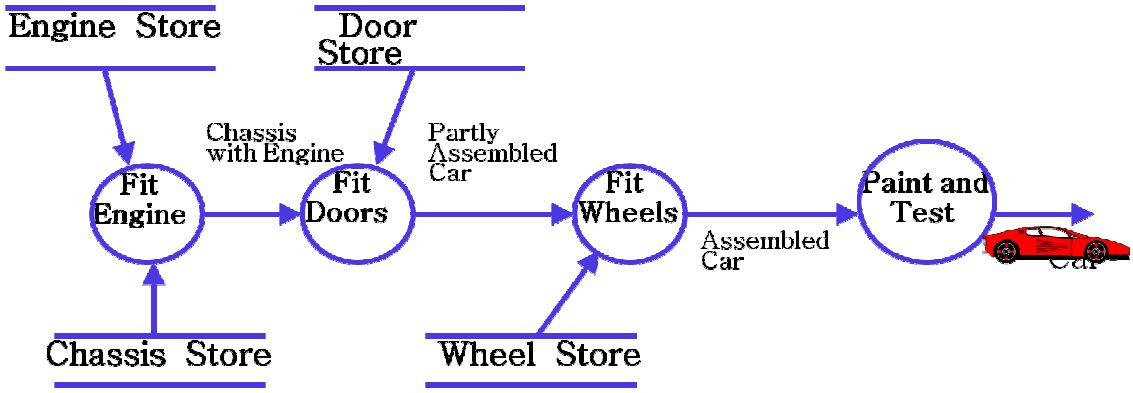
Data flow diagrams

- DFD is an elegant modelling technique:
  - useful not only to represent the results of structured analysis
  - applicable to other areas also:
    - e.g. for showing the flow of documents or items in an organization,
- DFD technique is very popular because
  - it is simple to understand and use.
- DFD is a hierarchical graphical model:
  - shows the different functions (or processes) of the system
  - data interchange among the processes.

DFD Concepts

- It is useful to consider each function as a processing station:
  - each function consumes some input data and
  - produces some output data.

Data Flow Model of a Car Assembly Unit

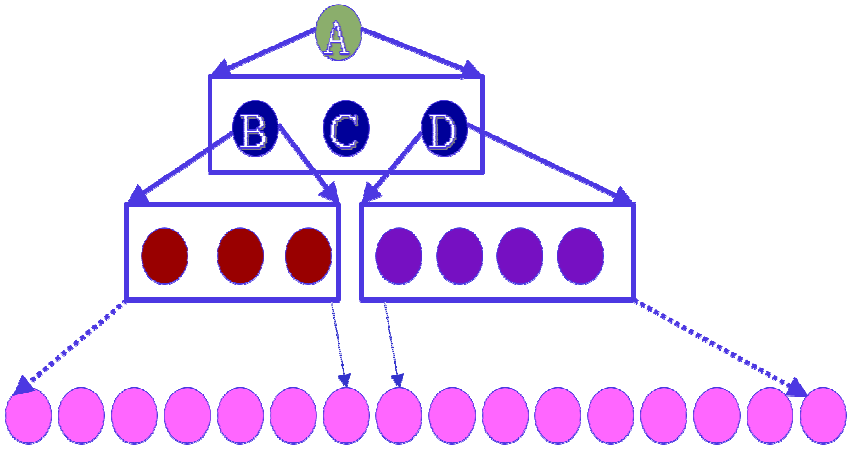


- A DFD model:
  - uses limited types of symbols.
  - simple set of rules
  - easy to understand:
    - it is a hierarchical model.

Hierarchical model

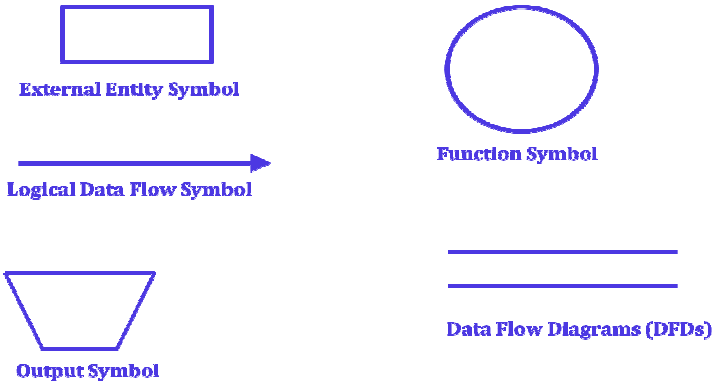
- Human mind can easily understand any hierarchical model:

- in a hierarchical model:
  - we start with a very simple and abstract model of a system,
  - details are slowly introduced through the hierarchies.



Data Store Symbol

⌘ **Primitive Symbols Used for Constructing DFDs:**



External Entity Symbol

Represented by a rectangle

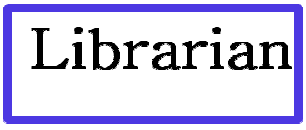
External entities are real

physical entities:

input data to the system or

consume data produced by the system.

Sometimes external entities are called terminator, source, or sink.



Function Symbol

A function such as “search-book” is represented using a circle:

This symbol is called a process or bubble or transform.

Bubbles are annotated with corresponding function names.

Functions represent some activity:

function names should be verbs.

Data Flow Symbol

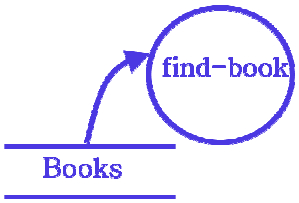
A directed arc or line.

represents data flow in the direction of the arrow.

Data flow symbols are annotated with names of data they carry.

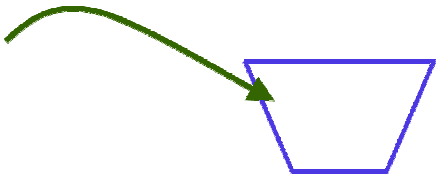
Data Store Symbol

- Represents a logical file:
  - A logical file can be:
    - a data structure
    - a physical file on disk.
  - Each data store is connected to a process:
    - by means of a data flow symbol.



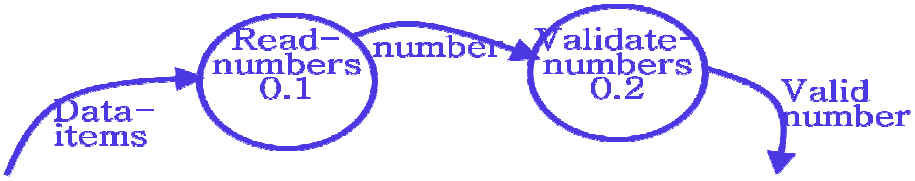
Output Symbol

Output produced by the system



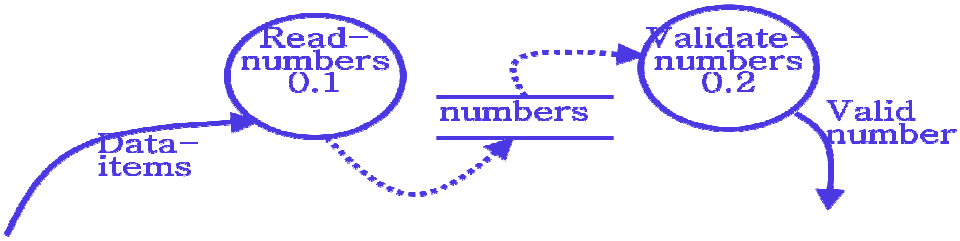
Synchronous operation

If two bubbles are directly connected by a data flow arrow: they are synchronous



Asynchronous operation

If two bubbles are connected via a data store: they are not synchronous.



How is Structured Analysis Performed?

- Initially represent the software at the most abstract level:
  - called the context diagram.
  - the entire system is represented as a single bubble,
  - this bubble is labelled according to the main function of the system.

Context Diagram

- A context diagram shows:
  - data input to the system,
  - output data generated by the system,
  - external entities.
- Context diagram captures:
  - various entities external to the system and interacting with it.
  - data flow occurring between the system and the external entities.
- The context diagram is also called as the level 0 DFD.
- establishes the context of the system, i.e. represents: