Unit-4

* Software Design and Implementation

Software design phase acts as the most creative phase in software development where designer plans regarding "how" a software system should be produced so that it is:

(i) functional
(ii) Reliable
(iii) Easy to understand
(iv) Easy to maintain & modify

A software requirement specification (SRS) document resulted from Requirement phase of software Development life cycle describes "what" a system does and acts as an input to the design process which describes "how" a software system works. Software system design deals with how requirement are realized. the output Design phase is software Design Document (SDD). SRS document becomes input design phase The main purpose of ~~software~~ design phase is to provide a solution of a problem given in SRS document.

→ Items Designed During Design Phase

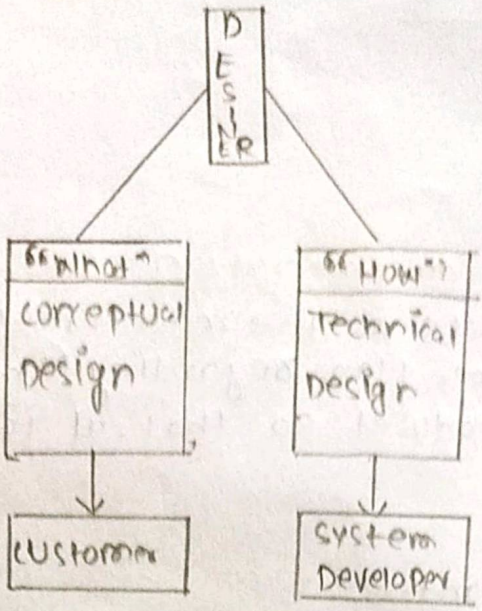
(i) Different modules which are required to implement the design solution.
(ii) control relationship among various modules.
(iii) Data structure of individual modules.
(iv) interface among different modules i.e, details of data items transferred among different modules.
(v) Algorithm required to implement the individual module.

Types of Design

(i) conceptual Design / High level design

(ii) Technical Design / Detailed Design

(ans)



*. (i) conceptual Design :- The intial stage of design phase is the conceptual design. conceptual design provides information to the costomer regarding what the system will do.

conceptual design answer the following question

(i) What will be the data source?
(ii) what will happen to data in the system?
(iii) What will be the system view to users?
(iv) what is timing of events?
(v) what will be the structure of forms & reports?

(ii) Technical Design :- once the conceptual design is approved by the costomer, it is translated into a more detailed document known as Technical design. The technical design allows the system developers to understand the actual hardware & software requirement to solve the costomer's Problem. In addition the technical design describes the hardware configuration the software needs, the communication interfaces, the input and output of system, the network architecture and everything that translates the requirements into a solution to the costomer problem.

Advantages of Modularity

(i) System Debugging is easier.
(ii) Modularity helps in isolating the system problem to a component

Abstraction

Abstraction is a tool that permits a designer to consider a component at an abstract level without thinking about the details of the implementation of the component. An Abstraction of a component describes the external behavior of that component without bothering for internal details that produce the behaviour.

Information hiding

In this approach, each module in the system hides the internal details of its processing activities from other one. modules communicate only through well defined interfaces i.e, that implementation may change without effecting the module's client. provided that the interface remains unchanged.

→ The design parts that can be hidden from other modules can be:
(a) A data structure and its implementation
(b) Character codes if any (used) and implementation
(c) masking, shifting of bits operation & other machine depended details.

concurrency

Software system can be categorized as sequential or concurrent. In sequential system, only one portion of the system, is active at any given time. concurrency system have independent process that can be activated simultaneously If multiple processors are available. on a single process concurrent process can be interleaved in execution time.

Aesthetics

Aesthetics means the artistic touch added to the design which includes simplicity, elegance and clarity These factors add to the pleasing appeal of software and lift its quality from mediocre status.

* Design concepts

There are some fundamental principles of software design which are used to develop the software.

→ Modularity
→ Abstraction
→ information Hiding
→ concurrency
→ Aesthetics

1. **Fundamental Modularity** :- software is divided into separately named & addressable components known as modules. that are integrated to satisfy the problem requirement.

* characteristics of modular system

(I) each module is well defined subsystem that used in various application.
(II) each module constitutes a single & well defined purpose.
(III) Modules can be compiled separately
(IV) Modules should be easier to use than to build
(V) Modules should be simpler from outside than from inside

* characteristics of effective modular system

(I) **Modular Decomposability** : If a design method provides a mechanism for decomposing the problem into subproblem, it reduces the complexity of the problem resulting in an effective modular system.

(II) **Modular composability** : If a design method enables the existing design components to be assembled into a New system then it results an effective modular system

(III) **Modular understandability** : if a module is understandable as a standalone unit without reference to other Modules, it is easier to build & easier to change.

(IV) **Modular continuity** :- if small changes to the system requirements changes the individual modules rather than changing the complete system then the change-induced effects are minimized.

(V) **Modular protection** :- if an unrequired condition occurs with in a module & its effects are limited that module only then the effects of change induced are minimized.

*STRUCTURED SOFTWARE DESIGN TECHNIQUES

Software design is important activity of software engineering. The outcome of a bad design is very hard to handle. Hence a best software strategy and method that suits a particular application should be used. Designed must be talented, intuitive and creative. A design that can be understood, copied & modified is expensive to maintain.

software design is the first part of ~~the~~ three technical activities i.e, design, code generation and testing that are required to build and verify the software.

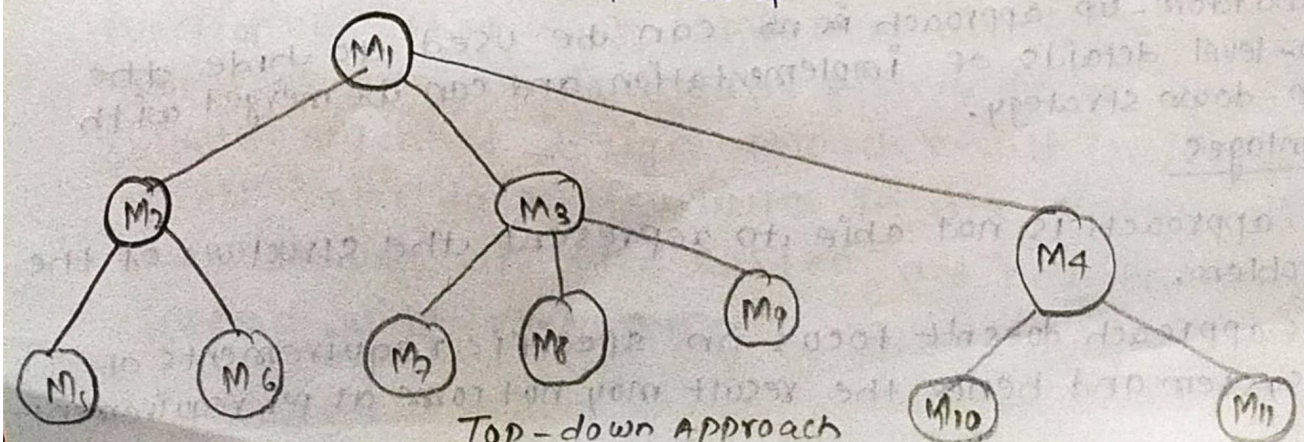
→To design such a hierarchy, the following approaches are possible

(I) Top-down design strategy
(II) Bottom-up design strategy

(I) Top-down design strategy :- A Top-down design approach consists of identifying the major components of the system decomposing them into their lower-level components and repeating it untill the desired level of detail is achieved. Starting from an abstract design, the design is refined to a more concrete level in each step, untill a level is reached where no more refinement is required & the design can be implemented directly.

*Objectives of Top-down design

(I) To systemise the design process
(II) To produce a modular program design
(III) To provide a framework in which a problem can be effectively solved



Top-down Approach

Advantages

(i) Top-down design strategy strongly focuses on specific requirements which help to make a design responsive to its requirements.

Disadvantages

(i) In Top-down design approach system boundaries are application as specification dependent. Thus the components can't be reused.

(ii) If coding of a part starts soon after it's design, it is impossible to test anything untill all the subordinate modules are coded.

(ii) Bottom-up design strategy:- Bottom-up design approach starts with designing the most basic or primitive components and proceeds to higher-level components that use these lower-level components. initially those modules are identified that are required by many programs. These modules are then collected are then collected together in the form of a 'library'. These modules may be for mathematical functions, for input-output function or for graphical functions.

The approach represents that style of design where modules are combined to provide large modules and then to combine those to provide even large modules and so on till one big module is produced which represents the complete desired program.

Advantages

(i) Bottom-up design strategy is an economical design approach as the possible solutions can be reused.

(ii) To bottom-up approach ~~is no~~ can be used to hide the low-level details of implementation and can be merged with top-down strategy.

Disadvantages

(i) This approach is not able to represent the structure of the Problem.

(ii) This approach doesn't focus on specific requirements of the system and hence the result may not come as per requirement

* Software Design Method

The design process starts when the requirement document (SRS) for the software to be developed is available. The design process for software system constitute two levels. At the first level, the modules which are required for the system are identified. In addition the specifications of these modules and the way of interconnection b/w modules is also considered. This process is known as a system-design.
A design-methodology, is a systematic approach for creating a design by using a set of techniques and guidelines. Design methodology are supposed to focus on the system design. The design methodology that can be used by the developer to design system.

→ Software-design methods can be of two forms

(I) function-oriented
(II) object-oriented

(I) function-oriented

function oriented design is an approach to software design where design is decomposed into a set of interacting units where each unit has a clearly defined function. Thus system from a function view-point.
In function-oriented design the design consist of module definitions and each module supports a function abstraction. In this design approach system is viewed as a transformation function which transforms the input to the desired output. The main purpose of the design-phase is to specify the components a transformation function so that each component also represents a transformation function. each system design phase using function oriented design approach result in definition of all major data structures in the system, all major modules of the system and how the modules interact with each other.

(ii) Object-oriented design

Object-oriented technologies reflect a natural view of the world, objects are categorised into classes and class hierarchies.

→ each class contains a set of attributes that describe it and a set of operation define its behaviour.
→ But abject encapsulate both data & process.
→ messages are passed to objects for initiating a processing operation.

+ There are three concepts which differentiate object oriented design

(a) Encapsulation, enables the attributes and operations of a class to be inherited by subclasses.
(b) Encapsulation, package data and the operations into single named abject.
(c) polymorphism, enables a number of different operation to have same name thereby reducing line of codes.

The o-oproducts and system evalve iteratively so that the final product will be developed over a series of increments.

* object-oriented vs function-oriented design

(i) Unlike function oriented design method, in OPD, the basic abstractions are not real world function such as sort display, track etc. but real world entities such as employee picture, mechine, radar system etc.

(ii) In abject oriented design, state information is not represented in a centralized shared memory but is distributed among the abjects of the system. for ex:- In an employee payroll system, the employee data such as names of the employee, their code numbers, basic salaries etc. are usally implemented as global data in a traditional programming system. Whereas in an objects oriented system these data are distributed among different employee abject of the system. object communicate by message passing. so, one abject may discover the state information of anthor abject by programming. it.

* Design Notation

In software design, the representation schemes are of prime-importance. good natation can clarify the Interrelationship and interaction of interest while poor notation can complicate the matter and can interfere with good design practice.

Types of Design specification

(i) External Design specification:- It describes the ~~control flow~~ external characteristics of a software.

(ii) Architectural Design specification:- It describes the structure of the system.

(iii) Detailed Design specification:- it describes the control flow, data representation and other algorithmic details within the modules.

During the design phase two ~~thing~~ things are prime interest namely: The design of the system which is basic, objective of this phase and the process of designing itself while designing, a designer needs to record his thoughts and decisions and to represent the design in a systematic way. for this very purpose design notation are used.

Example

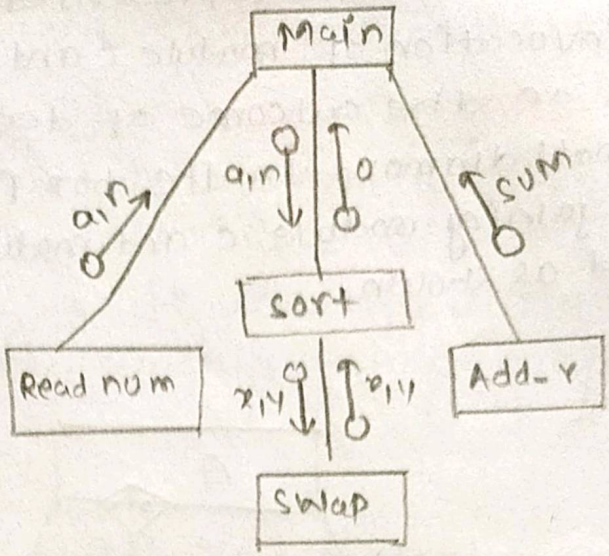
(i) Data flow Diagram (DFD)
(ii) Data Dictionary
(iii) Structure chart
(iv) Pseudo code
(v) Structure English
(vi) structure flow chart

* Structure chart

structure chart is used to graphically represent the function-oriented design. the structure of a program is made up of the modules of that program together with the interconnections between modules.

The following program of structure

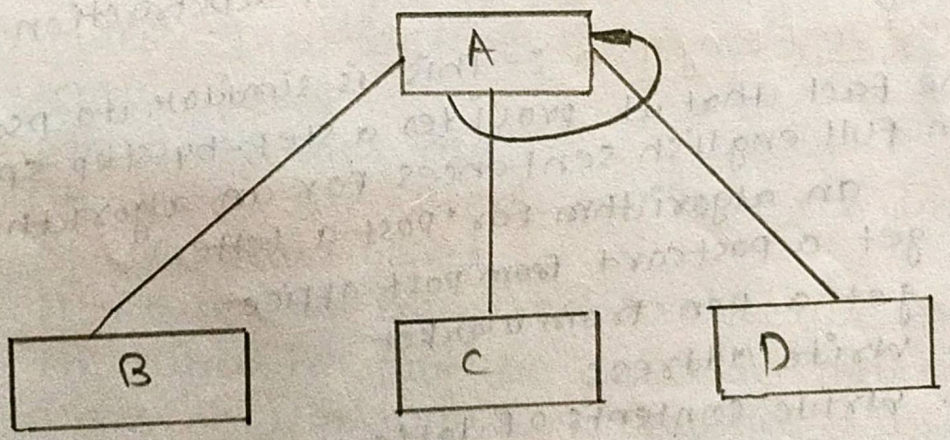
```
main()
{ int sum, n, N, a[max];
  readnum(a, &N);
  sort(a, N);
  scanf("%d", &n);
  sum = add_n(a, n);
  printf("sum");
}
readnum(a, N)
 int a[], *N;
{
   ~
   ~
}
sort(a, N)
int a[], N;
{ -
  if(a[i] > a[t]
   swap[a[i], a[t]);
   -
  }
add_n(a, n)
 int a[], n;
 { - -
   - -
 }
swap(float *a, float *b)
{ float temp = *a
        *a = *b
        *b = temp;
    }
```



Structure chart of SORT program

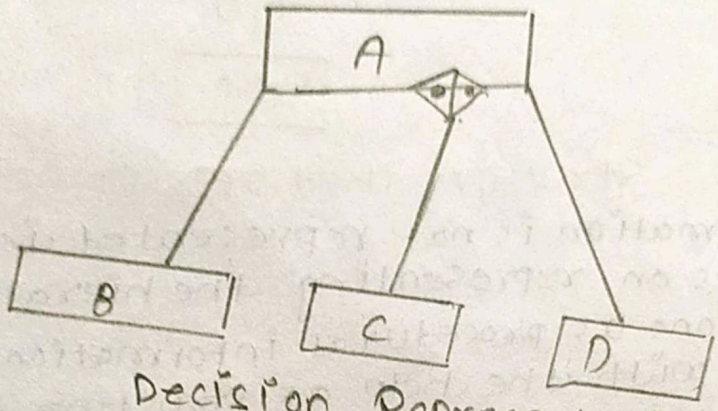
Procedual information is not represented in structure chart and the focuces on representing the hieraehy of modules. Loop & decisions as procedural information can also be represented with the help of structure chart.

e.g. if a module A has subordinates B, C, D and module A repeatedly calls the module C and module D. This can be represented by looping arrow around the arrows joining the subordinates C and D to A as shown.



Iteration Representation

Decisions can also be represented by structure chart e.g if the invocation of module c and module D in module A depends on the outcome of decision i.e, represented by a small diamond in the box for module. A with arrows joining module c and module D coming out of diamond as shown.



Decision Representation

(iv) Pseudo code :- In pseudo codes the designer describes in his own language using short phrases I keywords like. if-then-else, while-Do. These are another method of representing algorithm. The pseudocodes can be implemented to any level of abstraction for describing a system.

(v) Structured English :- This is similar to pseudocode but the fact that it provides a step-by-step specification in full english sentences for an algorithm.

ex :- an algorithm for "post a letter"

1. get a postcard from post office
2. get a pen from market
3. write address
4. write contents of letter
5. write until done
6. post it in red letterbox.

- What are the relationships between objects and the processes that transforms them.

9 ENTITY RELATIONSHIP DIAGRAM (ERD)

The entity Relationship Diagram (ERD) is the means of providing a data model of the system in structure analysis. It identify data objects and their relationship using graphical notation.

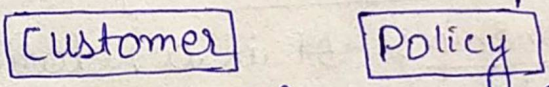
Components of ERD

E-R diagram is a detailed logical erepresentation of the data for an organization and constitute the following components:

(i) Data Entities / Data objects
(ii) Attributes
(iii) Relationship:

(i) Entities:- An entity is a data object about which information i.e the data object.

* Entities are shown in capital letters in E-R diagram.



Entities type in E-R diagram.

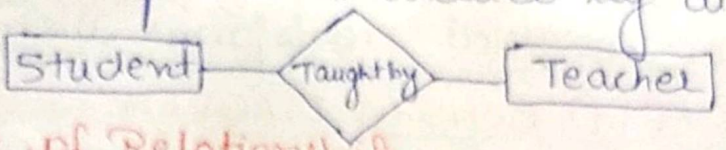
(ii) Attributes:- They define the properties of a data object

Their Characteristics are:-

(i) Attributes are used to name an instance of the data object.
(ii) Attribute can be used to describe the instance.
(iii) An attribute can be used to make reference to another instance in another table.

(iii) Relationships:- Data objects are connected to each other with the help of a relation. Relationship is used for associating two entity types.

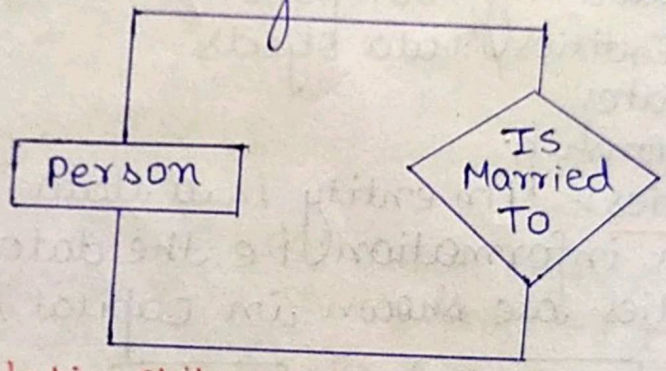
Relationship are indicated by diamond symbol.



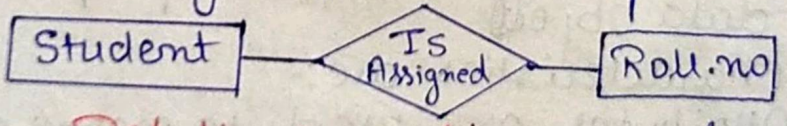
9.2 Degree of Relationship

The degree of relationship is defined as the number of entity types that participate in that relationship. Depending upon the number of entity types relationships in E-R model are classified as:-

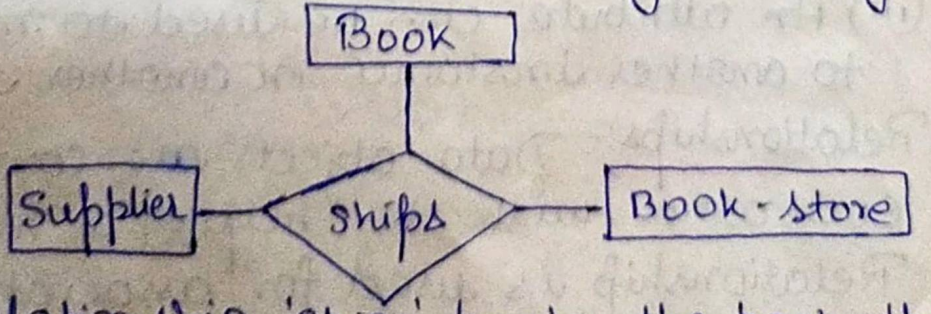
(i) **Unary Relationship** → It describes relationship between instances of one entity type. An instance is a single occurance of an entity type.



(ii) **Binary Relationship** :- It is a relationship between instance of two entity types and is most common type of relationship



(iii) **Ternary Relationship**: It is a relationship that describes relationship among 3 intity types.



The relationship 'ships' tracks the book that is shipped by a particular supplier to selected Book-store.

9.3 Cardinality

Cardinality is defined as the specification of the number of occurences of an object that can be related to the number of occurances of another object.

(1) One-to-one (1:1) In this type of relationship an occurance of object A can relate to one and only one occurance of object B and an occurance of ojject B can relate to only one occurance of object A.

(2) One-to-many (1:N) One occurance of an object A can relate to one or many occurance of object B but an occurance of object B can relate to only one occurance of object A.

(3) Many-to-Many (M:N) An occurrence of object A can relate to one or more occurrence of object B while an occurrence of object B can relate to one or more occurrence of object A.

10. DECISION TABLE

Decision table provides a mechanism for specifying complex decision logic. A decision table constitutes four basic elements namely:-

(i) Condition Stub:- It contains all the conditions being examined.

(ii) Condition Entries:- These are used to combine conditions into decision rules.

(iii) Action Stub:- It describes the actions to be taken in response to decision rules.

(iv) Action Entries :- These enteries relate decision rules to actions.