UNIT - III

Analysis model & Design

* Analysis Model :-

→ Analysis Model operates as a link between the "system description" and the "design model".

→ In the Analysis Model, Information, functions and the behaviour of the system is defined and these are translated into the architecture, interface and component level design in the 'design modeling'.

→ The Analysis Model actually a set of models, is the first technical representation of a system.

Goals of Analysis Model:-

- The products of analysis must be highly maintainable. This applies particularly to the target Document (S/w Requirements Specification (SRS))
- problem of size must be dealt with using an effective method of partitioning. The victorian novel specification is best.
- Graphics have to be used whenever possible.
- we have to differentiate between logic (Essential) and physical (implementation) considerations.

① Requirements Analysis :-

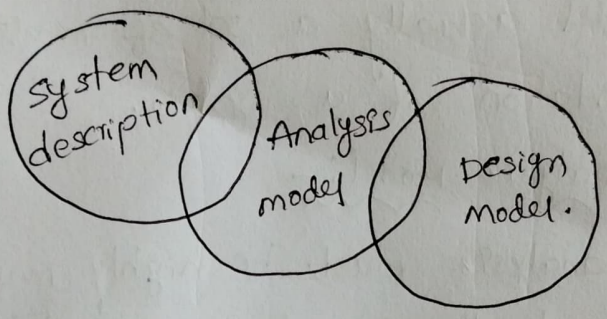
→ Requirements analysis results in a specification of S/w operational ~~characteristics~~ characteristics.

→ Requirements analysis allows the S/w Engineer to elaborate on basic requirements established during earlier requirement Engineering tasks & build models.

→ Requirements analysis provides the slw designer with a representation of information, function, & behavior that can be translated to architectural, interface & component-level designs.

→ finally requirements speafication provides the developer & the customer with the means to assess quality once slw is build.

→ The customer may be unsure of precisely what is required. The developer may be unsure that specific approach will properly accomplish function performance.



2) Overall objectives & philosophy:-

The analysis model must achieve three primary objectives.

1. to describe the what ~~we~~ the customer required.
2. to establish a basis for the creation of a slw design.
3. to define a set of requirements that can be validated once the slw is build.

. The analysis model bridges the gap blw system-level description that describes overall system functionality as it is achieved by applying slw, hlw, data, human, & other system elements.

. The slw design that describes the slw application architecture, user interface, & component level structure.

(3) Analysis Rules of Thumb :-

Arlow & Neustadt suggest a number of worthwhile rules of thumb that should be followed when creating the analysis model.

(a) The model should focus on requirements that are visible within the problem or business domain. The level of abstraction should be relatively high. "Don't get bogged down in details" that try to explain how the system will work.

(b) Each element of the analysis model should add to an overall understanding of sw requirements & provide insight into the information domain, function, & behavior of the system.

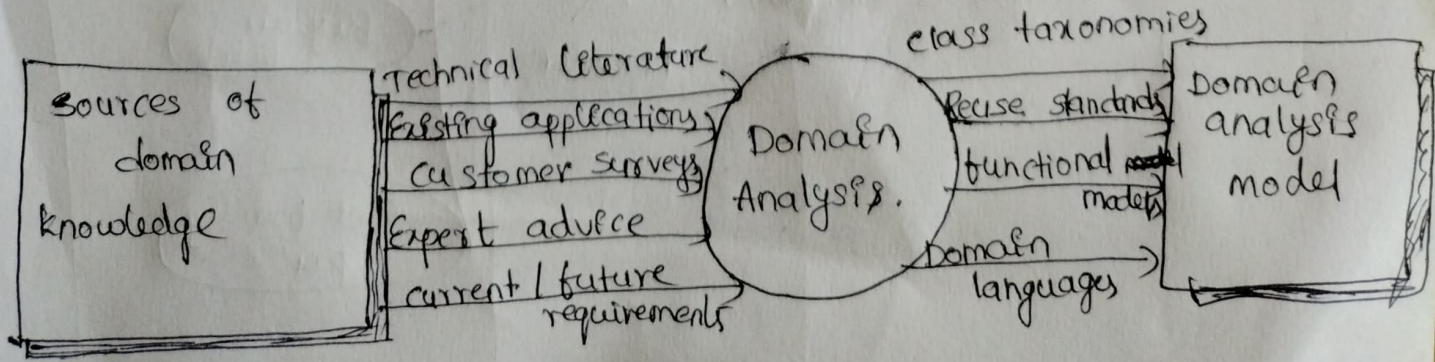
(c) Delay consideration of infrastructure & other non-functional models until design.

d) Minimize coupling throughout the system.

e) Be certain that the analysis model provides value to all stakeholders.

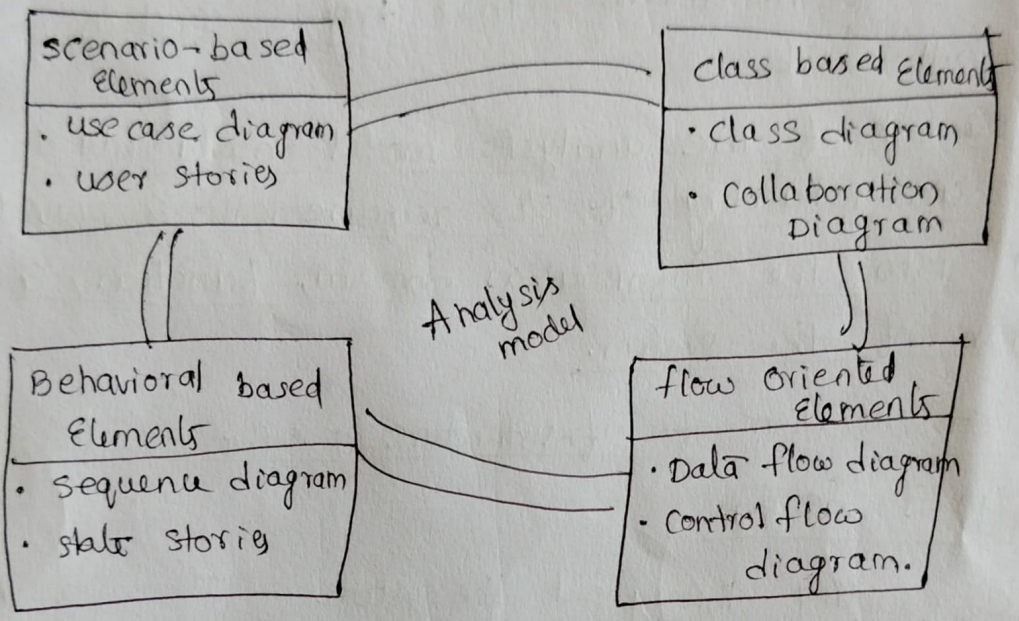
f) Keep the model as simple as it can be.

(4) Domain Analysis :-



The "specific application domain (SAD)" can range from avionics to banking, from ~~multiple~~ multimedia video games to slw embedded within medical devices. The goal of domain Analysis is straight-forward, to find or create those analysis classes and/or common functions & features that are broadly applicable, so that they may be reused.

Elements of the analysis model



Elements of analysis model.

(1) Scenario ~~based~~ based elements

- This type of elements represents the system user point of view.
- Scenario based elements are use case diagram, user stories

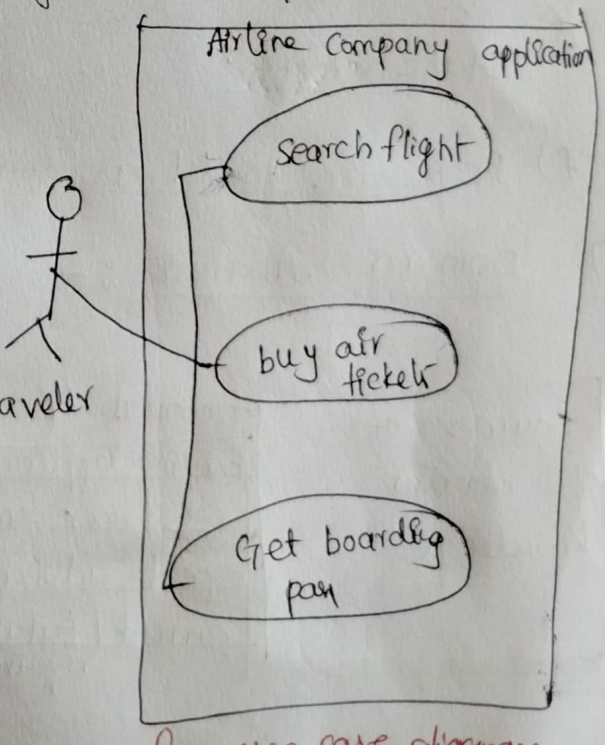


fig: use case diagram

(2) Class based Elements

- The object of this type of element manipulated by the system.
- It defines the object, attributes & relationship.
- The collaboration is occuring between the classes.
- Class based elements are the class diagram, collaboration diagram.

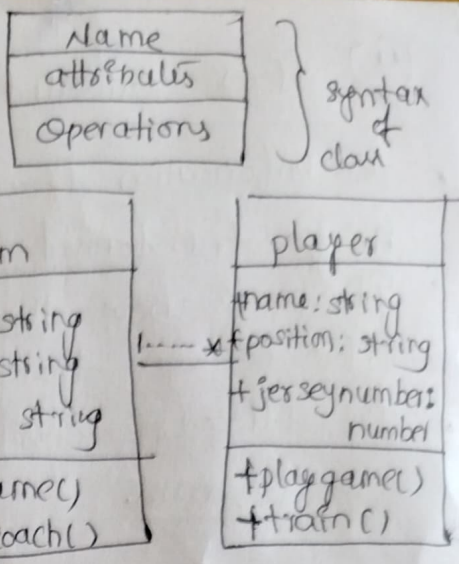


fig: class diagram

(3) Behavioral Elements :-

* Behavioral elements represent state of the system & how it is changed by the external events.
* The behavioral elements are sequence diagram, state diagram

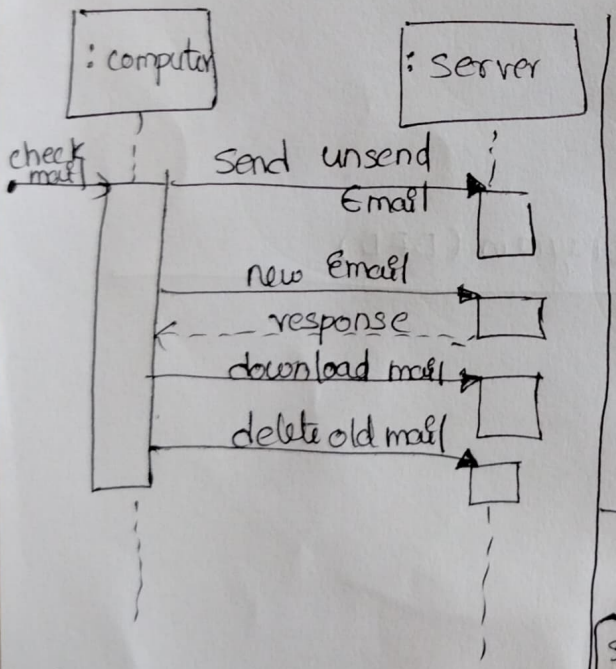
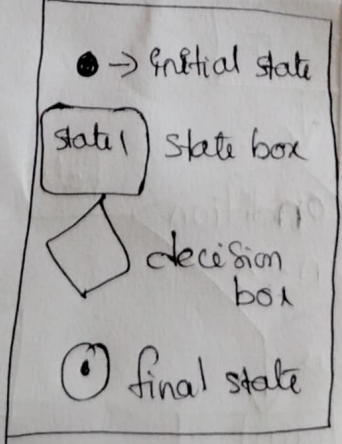


fig: sequence diagram



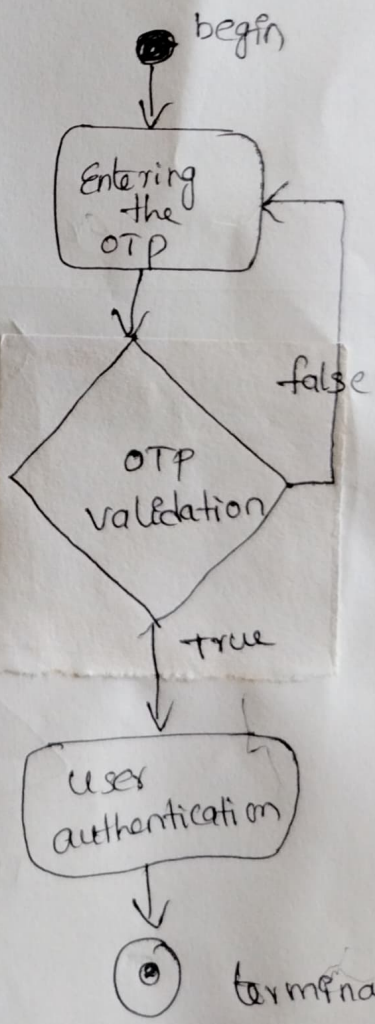
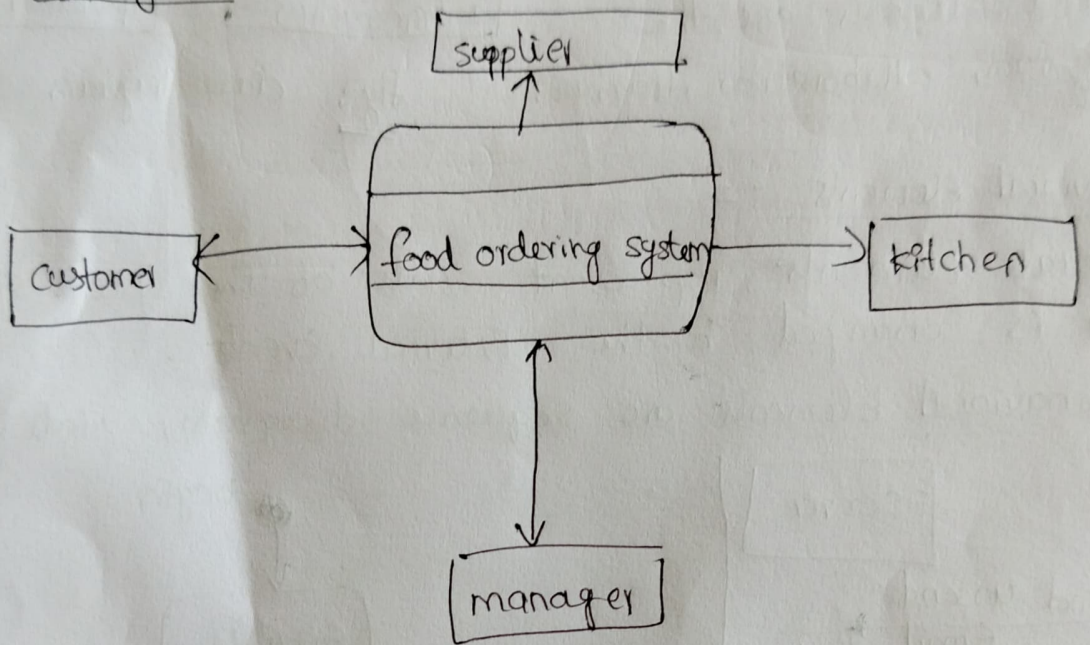


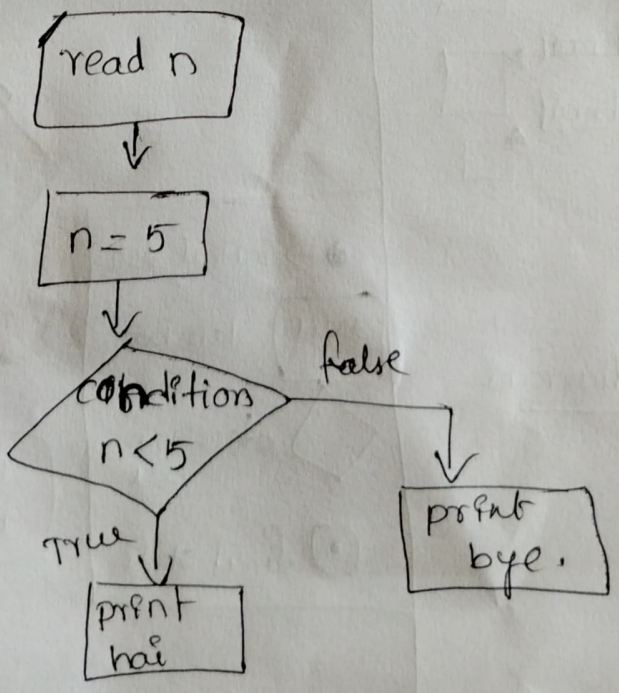
fig:- State diagram

(4) flow oriented Elements :-

- An information flows through a computer based system it gets transformed
- It shows how the data objects are transformed while they flow between the various system functions.
- The flow elements are data flow diagram, control flow diagram.



Ex Data flow Diagram (DFD)



Control flow diagram (CFD)

(6) (7)

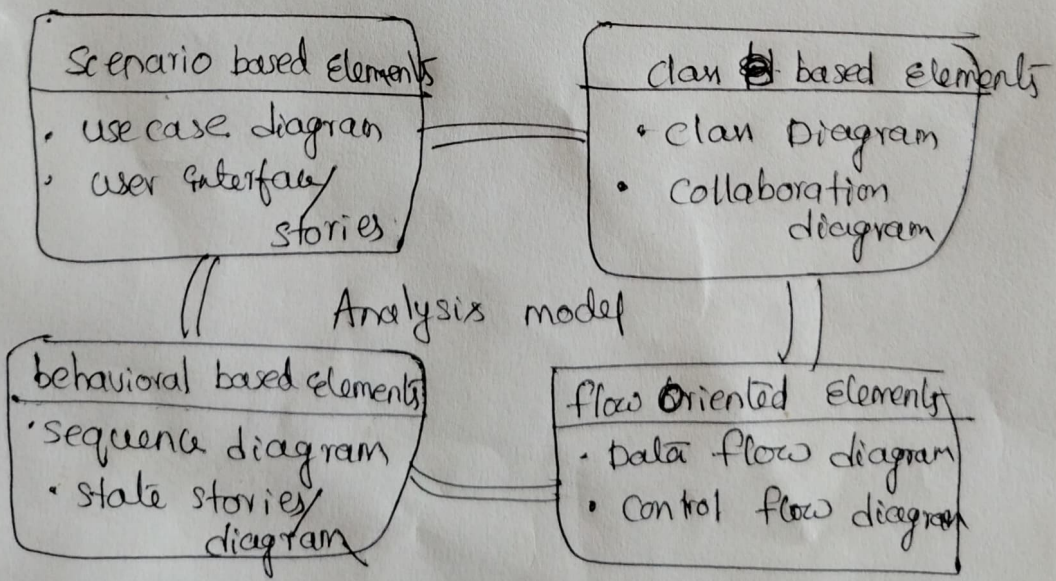
* Analysis modeling Approaches :-

→ One view of analysis modeling, is called structured analysis. Data Objects are modeled in a way that defines their attributes & relationships.

first approach :- process the manipulate data Objects are modeled in a manner that show how they transform data as data Objects flow through the system.

second Approach :- second approach to analysis modeling called Object oriented analysis.

Analysis modeling leads to the derivation of each of the modeling elements, the specific content of each element may differ from Object to project. The s/w team must work to keep it simple. Only those modeling elements that add value to the model should be used.



Elements of analysis model :-

refer previous concept (Analysis model)

UNIT III(PART-II)

Analysis Model and Design: Analysis model, Analysis modeling approaches, Data modeling concepts, Design process, Design quality, Design concepts.

➢ CONCEPTS OF DATA MODELING

- Analysis modelling starts with the data modelling.
- The software engineer defines all the data object that proceeds within the system and the relationship between data objects are identified.

Data objects

- The data object is the representation of composite information.
- The composite information means an object has a number of different properties or attribute.

  For example, Height is a single value so it is not a valid data object, but dimensions contain the height, the width and depth these are defined as an object.

Data Attributes

Each of the data object has a set of attributes.

Data object has the following characteristics:

- Name an instance of the data object.
- Describe the instance.
- Refer to another instance in another table.

Relationship

Relationship shows the relationship between data objects and how they are related to each other.

Cardinality

Cardinality state the number of events of one object related to the number of events of another object.

The cardinality expressed as:

One to one (1:1)

One event of an object is related to one event of another object.
For example, one employee has only one ID.

One to many (1:N)

One event of an object is related to many events.
For example, One collage has many departments.

Many to many(M:N)
Many events of one object are related to many events of another object.
For example, many customer place order for many products.

Modality

- If an event relationship is an optional then the modality of relationship is zero.
- If an event of relationship is compulsory then modality of relationship is one.

➢ DESIGN PROCESS

The design phase of software development deals with transforming the customer requirements as described in the SRS documents into a form implementable using a programming language. The software design process can be divided into the following three levels of phases of design:

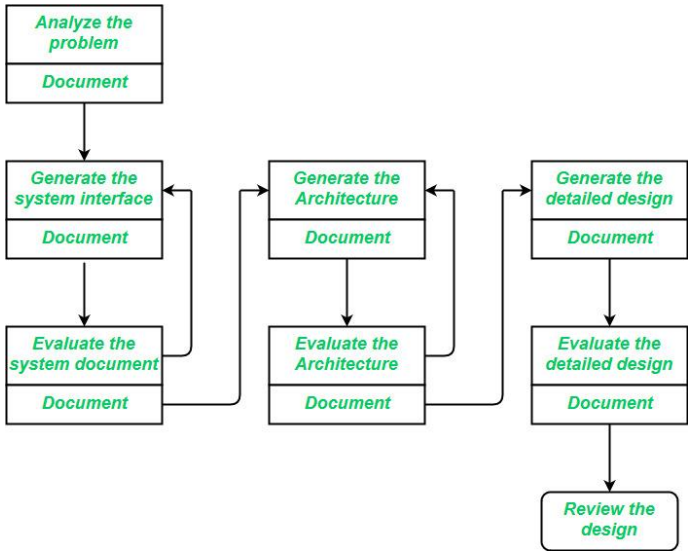
1. Interface Design

2. Architectural Design

3. Detailed Design

Elements of a System:

1. **Architecture –** This is the conceptual model that defines the structure, behavior, and views of a system. We can use flowcharts to represent and illustrate the architecture.
2. **Modules –** These are components that handle one specific task in a system. A combination of the modules makes up the system.
3. **Components –** This provides a particular function or group of related functions. They are made up of modules.
4. **Interfaces –** This is the shared boundary across which the components of a system exchange information and relate.
5. **Data –** This is the management of the information and data flow.



Interface Design: *Interface design* is the specification of the interaction between a system and its environment. this phase proceeds at a high level of abstraction with respect to the inner workings of the system i.e, during interface design, the internal of the systems are completely ignored and the system is treated as a black box. Attention is focused on the dialogue between the target system and the users, devices, and other systems with which it interacts. The design problem statement produced during the problem analysis step should identify the people, other systems, and devices which are collectively called *agents*. Interface design should include the following details:

- Precise description of events in the environment, or messages from agents to which the system must respond.
- Precise description of the events or messages that the system must produce.
- Specification of the data, and the formats of the data coming into and going out of the system.
- Specification of the ordering and timing relationships between incoming events or messages, and outgoing events or outputs.

Architectural Design: *Architectural design* is the specification of the major components of a system, their responsibilities, properties, interfaces, and the relationships and interactions between them. In architectural design, the overall structure of the system is chosen, but the internal details of major components are ignored. Issues in architectural design includes:

- Gross decomposition of the systems into major components.
- Allocation of functional responsibilities to components.
- Component Interfaces
- Component scaling and performance properties, resource consumption properties, reliability properties, and so forth.
- Communication and interaction between components.

The architectural design adds important details ignored during the interface design. Design of the internals of the major components is ignored until the last phase of the design.

Detailed Design: *Design* is the specification of the internal elements of all major system components, their properties, relationships, processing, and often their algorithms and the data structures. The detailed design may include:

- Decomposition of major system components into program units.
- Allocation of functional responsibilities to units.
- User interfaces
- Unit states and state changes
- Data and control interaction between units
- Data packaging and implementation, including issues of scope and visibility of program elements
- Algorithms and data structures

➢ DESIGN QUALITY

The main aim of design engineering is to generate a model which shows firmness, delight and commodity.

• Software design is an iterative process through which requirements are translated into the blueprint for building the software.

Software quality guidelines

• A design is generated using the recognizable architectural styles and compose a good design characteristic of components and it is implemented in evolutionary manner for testing.

• A design of the software must be modular i.e the software must be logically partitioned into elements.

• In design, the representation of data , architecture, interface and components should be distinct.

• A design must carry appropriate data structure and recognizable data patterns.

• Design components must show the independent functional characteristic.

• A design creates an interface that reduces the complexity of connections between the components. • A design must be derived using the repeatable method.

• The notations should be used in design which can effectively communicates its meaning.

Quality attributes

The attributes of design name as **'FURPS'** are as follows:

Functionality:

It evaluates the feature set and capabilities of the program.

Usability:

It is accessed by considering the factors such as human factor, overall aesthetics, consistency and documentation.

Reliability:

It is evaluated by measuring parameters like frequency and security of failure, output result accuracy, the mean-time-to-failure (MTTF), recovery from failure and the program predictability.

Performance:

It is measured by considering processing speed, response time, resource consumption, throughput, and efficiency.

Supportability:

- It combines the ability to extend the program, adaptability, serviceability. These three terms defines the maintainability.
- Testability, compatibility, and configurability are the terms using which a system can be easily installed and found the problem easily.
- Supportability also consists of more attributes such as compatibility, extensibility, fault tolerance, modularity, reusability, robustness, security, portability, scalability.

➢ DESIGN CONCEPTS

The set of fundamental software design concepts are as follows:

1. **Abstraction**

- A solution is stated in large terms using the language of the problem environment at the highest level abstraction.
- The lower level of abstraction provides a more detail description of the solution.
- A sequence of instruction that contain a specific and limited function refers in a procedural abstraction.
- A collection of data that describes a data object is a data abstraction.

2. **Architecture**

- The complete structure of the software is known as software architecture.
- Structure provides conceptual integrity for a system in a number of ways.

- The architecture is the structure of program modules where they interact with each other in a specialized way.
- The components use the structure of data.
- The aim of the software design is to obtain an architectural framework of a system.
- The more detailed design activities are conducted from the framework.

3. Patterns

A design pattern describes a design structure and that structure solves a particular design problem in a specified content.

4. Modularity

- A software is separately divided into name and addressable components. Sometime they are called as modules which integrate to satisfy the problem requirements.
- Modularity is the single attribute of a software that permits a program to be managed easily.

5. Information hiding

- Modules must be specified and designed so that the information like algorithm and data presented in a module is not accessible for other modules not requiring that information.

6. Functional independence

- The functional independence is the concept of separation and related to the concept of modularity, abstraction and information hiding.
- The functional independence is accessed using two criteria i.e Cohesion and coupling.

Cohesion

- Cohesion is an extension of the information hiding concept.
- A cohesive module performs a single task and it requires a small interaction with the other components in other parts of the program.

Coupling

Coupling is an indication of interconnection between modules in a structure of software.

7. Refinement

- Refinement is a top-down design approach.
- It is a process of elaboration.
- A program is established for refining levels of procedural details.
- A hierarchy is established by decomposing a statement of function in a stepwise manner till the programming language statement are reached.

8. Refactoring

- It is a reorganization technique which simplifies the design of components without changing its function behaviour.

- Refactoring is the process of changing the software system in a way that it does not change the external behaviour of the code still improves its internal structure.

9. Design classes

- The model of software is defined as a set of design classes.

- Every class describes the elements of problem domain and that focus on features of the problem which are user visible.