UNIT-II

- REQUIREMENTS ANALYSIS AND SPECIFICATION:
  - REQUIREMENTS GATHERING
  - REQUIREMENTS ANALYSIS
  - SOFTWARE REQUIREMENT SPECIFICATION (SRS)
  - FORMAL SYSTEM SPECIFICATION
- SOFTWARE DESIGN
  - OVERVIEW OF THE DESIGN PROCESS
  - HOW TO CHARACTERISE OF A DESIGN?
  - COHESION AND COUPLING
  - LAYERED ARRANGEMENT OF MODULES
  - APPROACHES TO SOFTWARE DESIGN

Introduction to Requirements Analysis and Specification:

→ Understanding the customer needs and expectations is extreamely important before moving toward designing a System.

→ The incomplete requirements and inconsistency requirements and incorrect requirements lead to failure of the projects.

→ Before going to develop the software product formal document is the most important.

→ The formal document is called "Software requirement Specification (SRS).

→ The SRS document is the basis of development of any product.

→ The SRS document consists of four categories.

1. User requirements
2. System requirements
3. Functional requirements
4. Non-functional requirements.

→ The change requirements is the important based on non-functional requirements change the functional requirements, based on the system requirements the User requirements is changed.

→ The change requirements and their informations are managed in a database or document.

✱ REQUIREMENT ENGINEERING PROCESS:

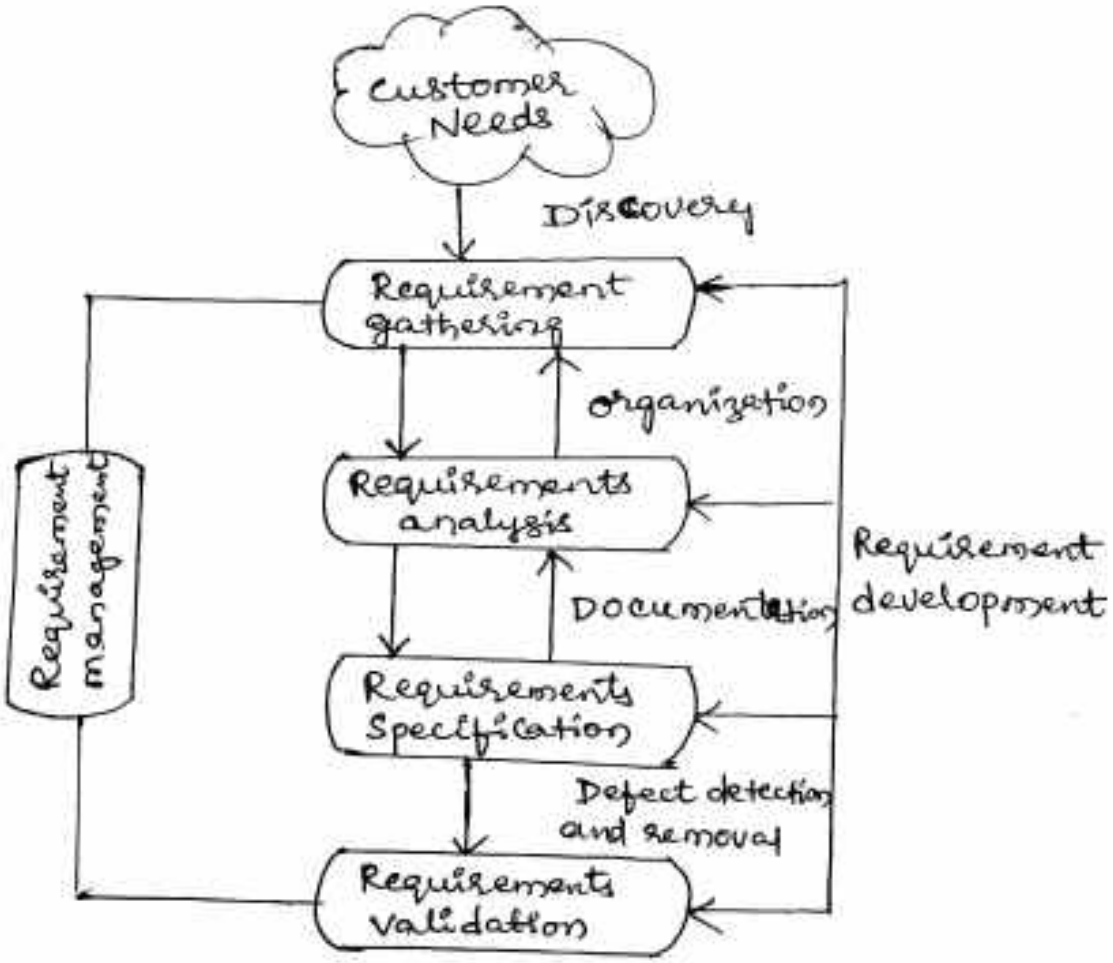


fig: Requirement engineering process.

→ Requirement engineering is the process of gathering, analysing documenting, validating and managed requirements.

→ The main goal of requirements engineering is to clearly understanding the customer requirements and systematically organize these requirements in the SRS.

→ Requirements engineering process begins once feasibility study and preliminary investigation are over.

→ It deals with the problem domain of the software that is to be converted into the solution domain.

In the requirement engineering process, the requirement elicitation phase aims to gather requirements from different perspectives to understand the customer needs.
It involves various stakeholders to elicit requirements and resolve social, technical, and communication issues.

Requirement Analysis concentrates on causes and effects, organization, and modelling of each requirents.

Requirement Specification is used to document the understood requirements that can be comunicated to the stakeholders. Finally,

Requirement validation activity ensures that all requirements are accurately specified, are complete and satisfy the customer needs.

REQUIREMENTS GATHERING:

→ Requirements gathering or requirement elicitation is the initial stage in the requirement engineering process for software development.

→ It is the problem understanding phase the existing system

→ The goals of requirement elicitation are to identify the different parties. The System analyst involves Software engineers, clients, endusers, Sponsors, managers, vendors and suppliers, stakeholders and follows Standards and guidelines to elicit requirements, shown in figure.

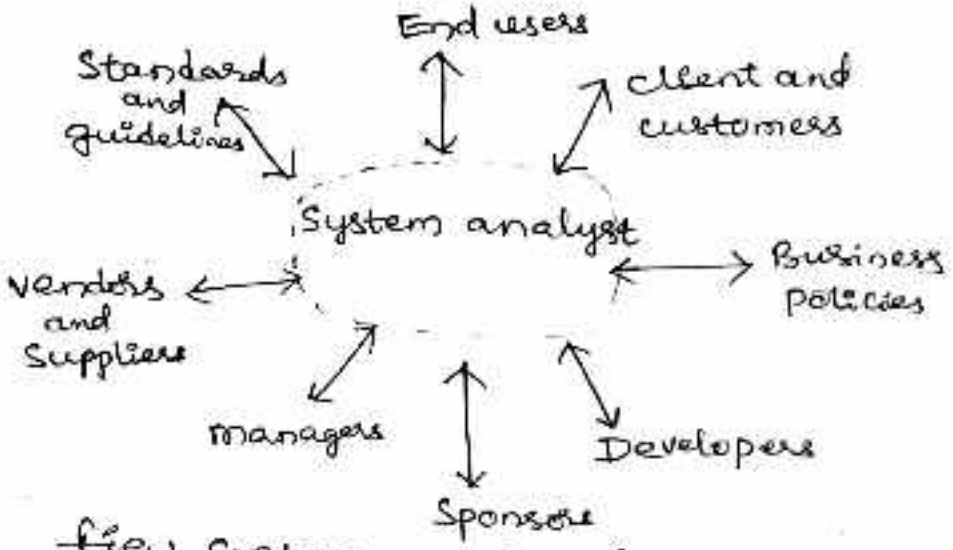


fig: System analyst interaction.

→ The role of the System analyst is multifunctional, and challenging as compared to other people in the organization.

→ Fact-finding techniques are used to discover User and requirements gathering.

Fact-finding Techniques:

1. Interviewing — face to face interaction with individual & group of people to the purpose of elicitating requirements.
2. Questionnaries — These are used to collect requirements at record
3. Joint Application development (JAD) — Workshops
4. Onsite observation — Analyst visit client site Organizations to collect data.
5. prototyping — It is the existed templates.
6. viewpoints — viewpoints are collected from different People stakeholder, Organizational.
7. Review Records — Reviewing the existing documents is a good way of gathering requirements.

REQUIREMENTS ANALYSIS:

→ Requirement analysis is the one of the phase of requirement engineering process.

→ After completion of requirements gathering, the requirement analysis is start.

→ In requirement analysis, we analyse the stakeholders needs, constraints and their information to the proposed system.

→ Requirement analysis models are prepared to analyse the requirements.

The following analysis techniques are used to modelling of requirements.

1. Structured analysis
2. Data-oriented analysis
3. Object-oriented analysis
4. prototyping.

Structured Analysis:

→ structured analysis is the one of the approach for requirement analysis.

→ DFD (Data flow) modelling is useful to understand the working process of the system.

→ The data is transformed into the model.

→ Structured analysis uses a graphical tool called "Data flow diagram".

→ DFD supports the data dictionary.

→ Data dictionary means data about data.

Data-Oriented analysis:

→ Data Oriented analysis is also refered to as "data oriented modelling".

→ Data model is the abstraction of the data structured required by a database rather than the operations on those data structure.

→ Data model is independent of hardware or software constraints that are used to implement the database.

→ Data models are composed of data entities, associations among different entities.

→ The functional models that describes how the data will be processed in the system.

→ Data Oriented analysis is performed using entity relationship modelling (ERM).

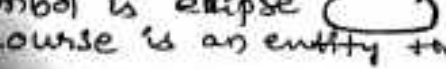
ERM (Entity Relationship modelling):

Entity relationship modelling is a pictorial method of data representation. It is a graphical manner.

Entities: Entity is an realworld object. An entity represents people, places or things. It is the data objects about which information is stored. Entity is denoted by the symbol is rectangle.

Example: Student, Employee, course, University, fees etc. These represents entities.

Attributes: Attributes are the properties of an entity. Each entity will have one or more data attributes. It is denoted by symbol is ellipse



course is an entity that attributes are course name, id, credits etc.

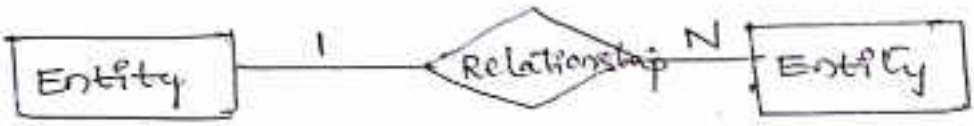
Relationships and cardinality

The relationship is indicated by ◇.
The relationship with two entities are

1. One-to-one
2. One-to-many
3. Many-to-One
4. Many-to-many

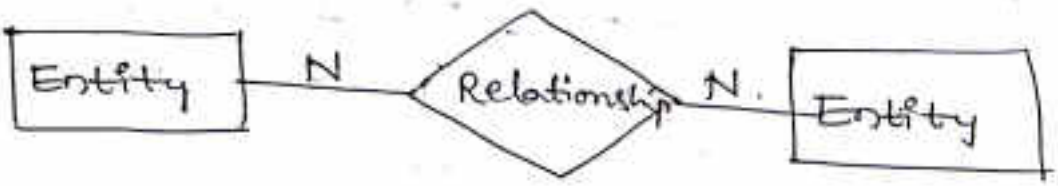
1. One-to-one

2. One-to-many

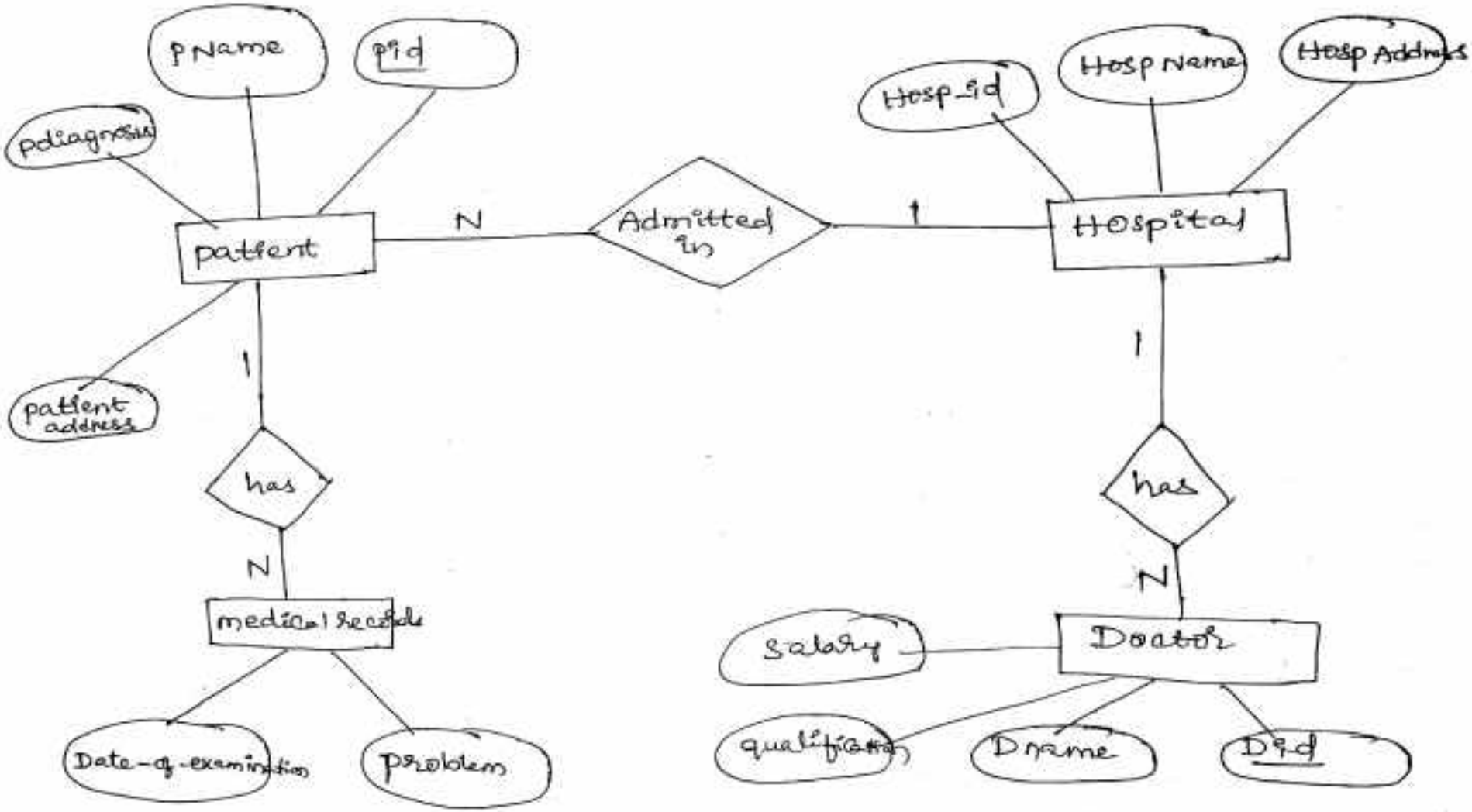


3. Many-to-one

4. Many-to-many



Example : E-R diagram for Hospital management System.



3. Object-Oriented analysis (OOA):

→ Object Oriented analysis (OOA) deals with the engineering requirements and modelling the application in the problem domain.

→ Object Oriented Design (OOD) translates OOA into programming constructs to produce practical design.

→ Object Oriented Analysis (OOA) increases the understanding of problem domain, it translates from the analysis to design phase.

→ OOA concentrate on ① object modeling ② Dynamic modeling

Objects and classes (Object modelling)

Objects are the real world entities that can uniquely be identified and distinguished from other objects.

Each Objects has certain attributes and operations.

Similar Objects are grouped together to form a class.

Classes are formed by identifying the common attributes common operations and common relationships in the similar objects.

Attributes are the data values of an Object.

Operations are services & functions of an object in a class.

Association The relationship between classes is known as "Association".

Class diagram association of employee in a company:

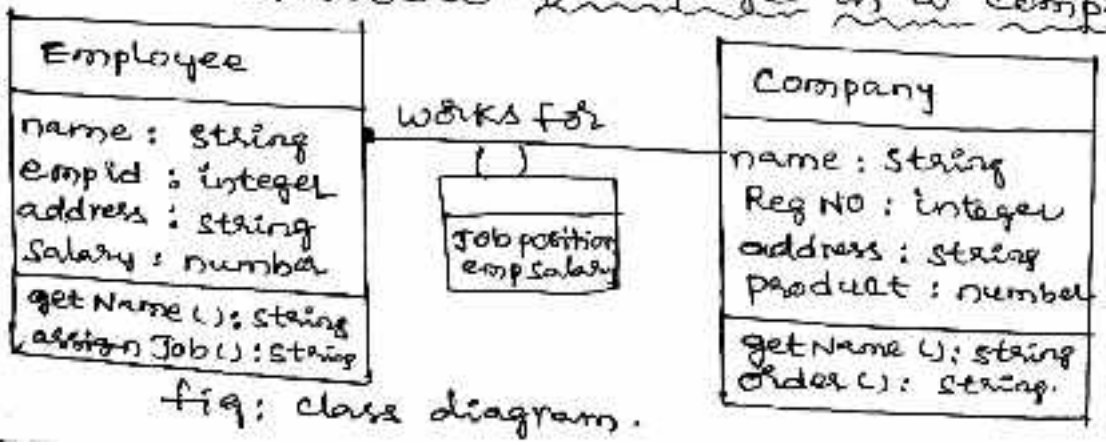


fig: class diagram.

Dynamic modelling

The dynamic model is the most suitable representation of the sequence diagram. The sequence diagram consists of entities and timeslots, it is a graphical tools of notations.

Ex: The ATM application is the best example for draw the sequence diagram.

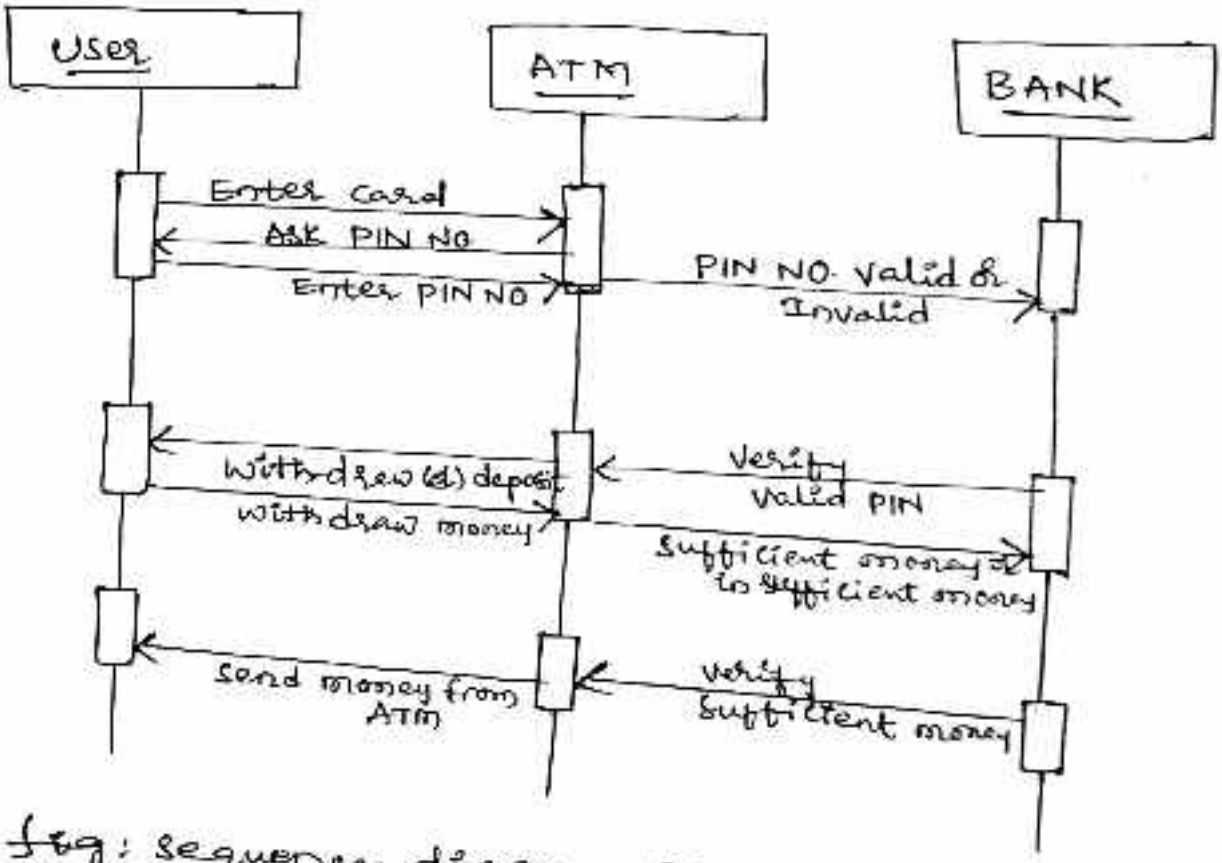


fig: Sequence diagram for ATM application.

Prototyping Analysis:

Traditional approaches, such as process oriented, data oriented, and Object oriented approaches, work well if all the requirements are known in advance and less user interaction is required for long term projects. These approaches have descriptive process for the representation of analysis outcomes and organisation of requirements.

Prototyping takes a different approaches for elicitation, analysis, and validation of the requirements.

This approach is more suitable where requirements are not known in advance, rapid delivery of the product is required.

There are two types of prototyping approaches.

1. Throwaway prototyping
2. Evolutionary prototyping.

1. Throwaway prototyping:

→ In this approach final prototype may be considered as the basis for writing the SRS document.

→ Firstly, the requirements are analysed and build prototype. that requirements properly working are not working identified & evaluate.

→ The requirements are properly working that requirements are converted into final system development Otherwise modify requirement. It is an iterative manner.

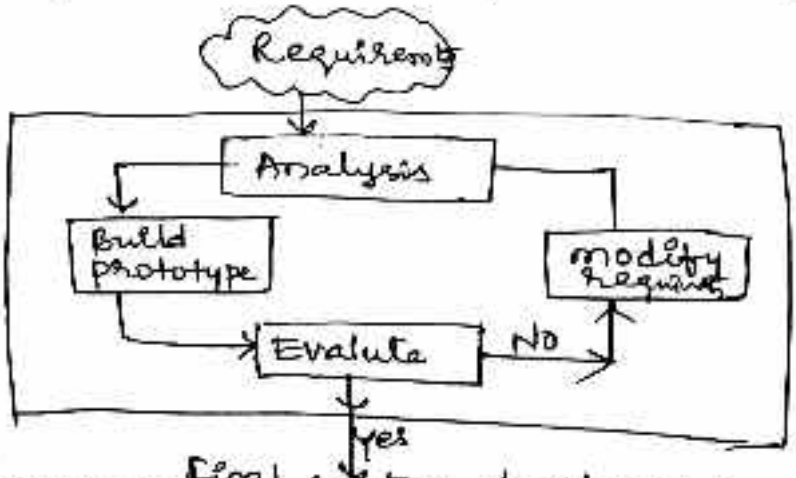
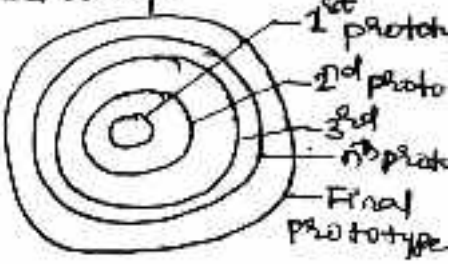


fig: Throwaway prototype.

2. Evolutionary prototype

It produced several iterations. The customer satisfies the requirements that will be maintaining iteratively.



SOFTWARE REQUIREMENT SPECIFICATION (SRS)

→ Software requirement specification (SRS) document is a formal document that provides the complete description of the proposed software.

→ Software requirements specification is one of the important documents required in the software development.

→ A good quality SRS ensures high quality software product.

Characteristics of the SRS:

1. Correctness
2. Unambiguity
3. Completeness
4. Consistency
5. Verifiability
6. Modifiability
7. Testability
8. Validity

Structure of SRS

The structure of an SRS describes the organization of the software requirement document.

The IEEE has published a standard for writing SRS.

SRS structure is shown in figure.

1. Introduction
   - 1.1 purpose
   - 1.2 Scope
   - 1.3 Definitions, abbreviations
   - 1.4 References
   - 1.5 Document Overview

2. General description
   - 2.1 product perspective
   - 2.2 product functions
   - 2.3 User characteristics
   - 2.4 General constraints
   - 2.5 Assumptions and dependencies

3. Specific requirements
   - 3.1 Functional requirements
     - 3.1.1 Item Introduction
     - 3.1.2 Input
     - 3.1.3 processing
     - 3.1.4 Output
   - 3.2 Non functional requirements
   - 3.3 External Interface requirements
   - 3.4 performance requirements
   - 3.5 Design constraints
   - 3.6 Security requirements
   - 3.7 Maintainability requirements
   - 3.8 Reliability requirements
   - 3.9 Availability requirements
   - 3.10 Database requirements
   - 3.11 Documentation requirements
   - 3.12 Safety requirements
   - 3.13 Operational requirements
   - 3.14 Site adaption

fig: IEEE standard for SRS.

Components of an SRS:

1. User requirements
2. System requirements
3. Functional requirements
4. Non-functional requirements.

User requirements:

→ User requirements are the high level abstract statements supplied by the customer, end users.

→ These requirements are the functionalities that the system is expected to provide within certain constraints.

→ These requirements are translated into system requirements.

→ User requirements are generally represented in natural language.

System requirements:

→ The system requirements are the detailed and technical functionalities written in a systematic manner that are implemented in the business process to achieve the goal of User requirements.

System requirements are functional and non-functional requirements.

Ex:
An ATM machine, User requirements allows user to withdraw and deposit cash.
The system requirements consider customer ID, Account type, bank name, PIN, communication link, Hardware, Software → ATM will service one customer at a time.

Functional requirements:

Functional requirements are the behaviour & functions that the system must support.

The User inputs processes & services that will be provided outcomes.

Ex: Business Application ATM,

Administrative tasks, transactions, cancellations, authentication authorization, external interfaces, audit tracking.

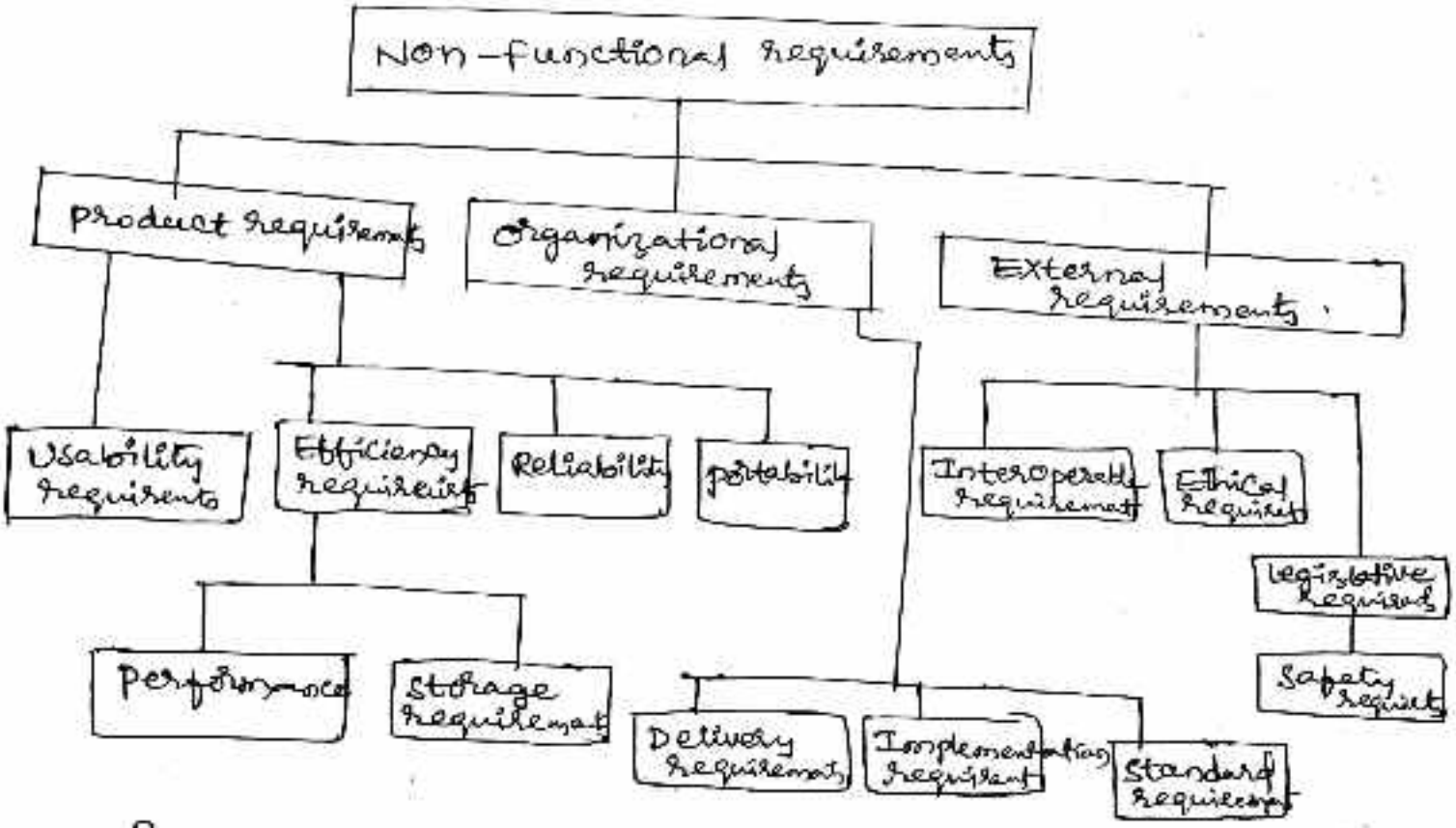
Non-functional requirements:

→ Non functional requirements are the product requirement organizational requirements, external requirements.

→ It is the properties of functional requirements.

→ It specifies how much fast, How much quality, how safety, secure security.

classifications of non-functional requirements.



fig; classifications of non functional requirements.

FORMAL SYSTEM SPECIFICATION:

→ mathematical approach can also be used to specify Software requirements.

→ mathematical representations include formal system specification, specification analysis and proof, transformational development, architectural specifications, program verification.

→ Formal methods can be used based on the application domain.

* Approaches to formal Specification:

There are various approaches to formal specifications.

1. Axiomatic Specifications
2. Model checking
3. program Annotations and tools.

Axiomatic specifications:

This approach defines the operations by logical assertions. Each operation has a set of pre-and post conditions. A predicate is specified as a Boolean expression which evaluates to true or false.

Ex: A student is promoted if he satisfies the credits

SPS = credits > 52 then promotes.

Model checking:

It is a technique for verification of finite-state systems.

Program Annotations and tools:

These are used to check behavioural properties of programs. Annotations can be represents as a logical formula of the specifications.

* Pros and Cons of formal methods

Formal methods uses short notations and allows using precise symbols and sentences.

· SOFTWARE DESIGN:

Introduction:

Software design is the process of describing the blueprint or sketch of the final software product in the form of design model. For example A Banking software system consists of various modules such as ATM interface, online transaction, loan management, deposit.

once the design is over, a design model is prepared and later it is translated into programming languages to produce the product.

. OVERVIEW OF THE DESIGN PROCESS:

→ A software design exists between requirement engineering and programming.

→ A software design process is a set of design activities carried out in the design phase to produce the design model from the SRS.

→ The design phases are

① Architectural design
② Physical design
③ Detailed design.

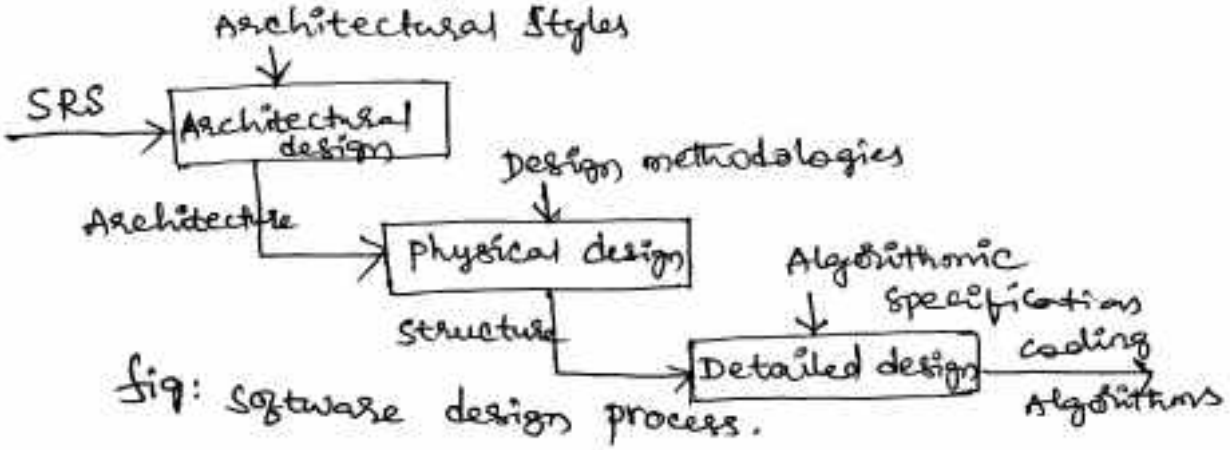


fig: Software design process.

Architectural design:

Architectural design is an external design, which describe the external behaviour of the software product. Such as Database, security, server maintaining, architectural styles are client server model.

physical design:

physical design is a high-level design or structural design which is concerned with refining the conceptual view of the systems.

Identifying the major modules, decomposing the modules into sub-modules, interconnections among the modules. This is called System design.

Detailed design:

Detailed design is the algorithmic design of each module in the software. It is also called "logical design". It concentrates on the specification of algorithms and data structure of each module.

The detailed design can be converted into source code.

CHARACTERISTICS OF A GOOD SOFTWARE DESIGN:

The quality of a software design can be characterized by the application domain. For example, real time Software will focus more on efficiency and reliability issues. The following good characteristics of software designs are.

1. correctness
2. Efficiency
3. Understandability
4. Maintainability
5. Simplicity.
6. completeness
7. Verifiability
8. portability
9. Modularity
10. Reliability
11. Reusability

1. correctness:

A design is said to be correct if it is correctly produced according to the stated requirements of customer in SRS.

2. Efficiency:

The efficiency of a design is concerned with performance related issues.

3. Understandability:

In a design, it should be easy to understand what the modules is, how it's connected to other modules,

4. Maintainability:

A difficult and complex design would take a larger time to be understood and modified.

5. Simplicity:

A Simple design will improve understandability and maintainability.

6. Completeness:

Completeness means that the design includes all the Specifications of the SRS.

7. Verifiability:

The design should be able to verified against the requirements documents and programs.

8. portability:

These architectures must be able to move a design to another environment.

9. Modularity:

A modular design will be easy to understand and modify.

10. Reliability: The reliability means within the time period the system don't failure.

11. Reusability: Use of reuse the components.

COHESION AND COUPLING:

Cohesion definition:

→ The modules or feenctions communicate within the single system is called as "cohesion".

→ Cohesion represents intra dependency between modules within the single system.

→ High cohesion maintaining into the project is most imptant.

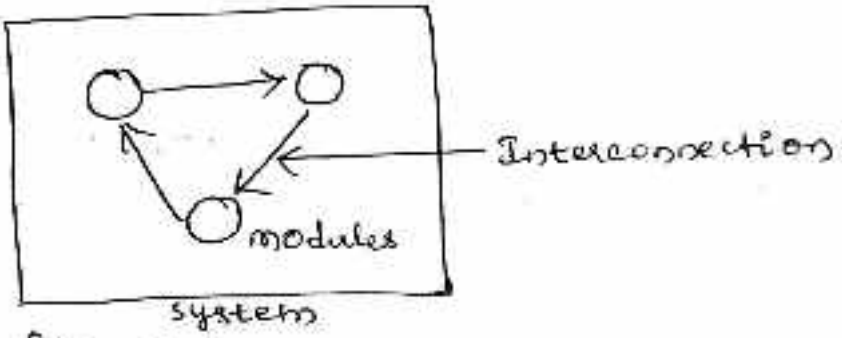


fig: Cohesion.

There are many different levels of cohesion. These are

1. Functional cohesion
2. Sequential cohesion
3. communicational cohesion
4. procedural cohesion
5. Temporal cohesion
6. Logical cohesion
7. coincidental cohesion.

1. Functional cohesion:

All the elements of the module perform a single function. Example mathematical functions are 'log' computes logarithm of a number and 'printf' prints the result.

2. Sequential cohesion:

Sequential cohesion exists when the output from one element of a module becomes the input for some other element. Example 'withdraw money' and update balance.

3. Communicational cohesion:

All the elements of a module operate on the same input & output data. Example 'print and punch the output file' can be communicational cohesion.

4. procedural cohesion:

It contains the elements that belong to a common procedural unit. Example entering, reading, and verifying the ATM password.

5. Temporal cohesion:

A module performs several functions in a sequence but their execution is related to a certain time. Ex example "a database trigger".

6. Logical cohesion:

Logical cohesion exists when logically-related elements of a module are placed together. These are input, error handling.

7. Coincidental cohesion:

It occurs when the elements within a given module have no meaningful relationship to each other.

Coupling definition

The modules & functions communicate between the systems is called "coupling".

→ coupling represents inter dependancy between modules of different systems.

→ Low coupling maintaining into the project is the most important.

→ In a coupling conflicts and collision issues are raised.

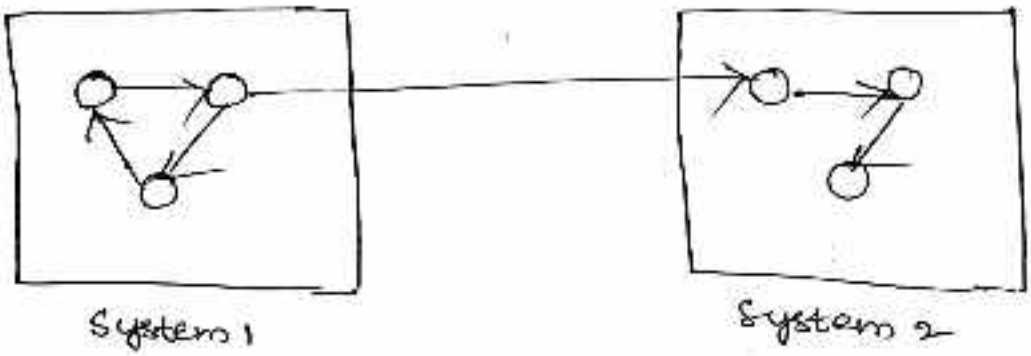


fig: coupling.

There are different types of coupling between modules such as

1. message coupling
2. Data coupling
3. Stamp coupling
4. Control coupling
5. External coupling
6. common coupling
7. content coupling.

1. Message coupling:

The Objects or components communicate through parameters or message passing.

EX: In c++ Objects communicate with each other by sending a message through parameter in the function call.

2. Data coupling:

Data coupling exists between modules when data are passed as parameters in the argument list of the function call.

3. Stamp coupling:

Stamp coupling occurs between modules when data are passed by parameter using complex data structure.

Ex: Structures in c.

4. control coupling:

Control coupling exists when one module controls the flow of another by passing control informations such as flag set or switch statements.

5. External coupling:

External coupling occurs when two modules share an externally imposed data format.

6. Common coupling:

Common coupling occurs when two modules share common data (EX: A global variable).

7. Content coupling:

Content coupling exists between two modules when one module refers or share its internal working with another module.

LAYERED ARRANGEMENT OF MODULES:

→ The layered architecture is composed of different layers.

→ Each layer is perform Specific Operations.

→ The Outer layer is responsible for performing the User interface operations while the components in the inner layer perform Operating system interfaces.

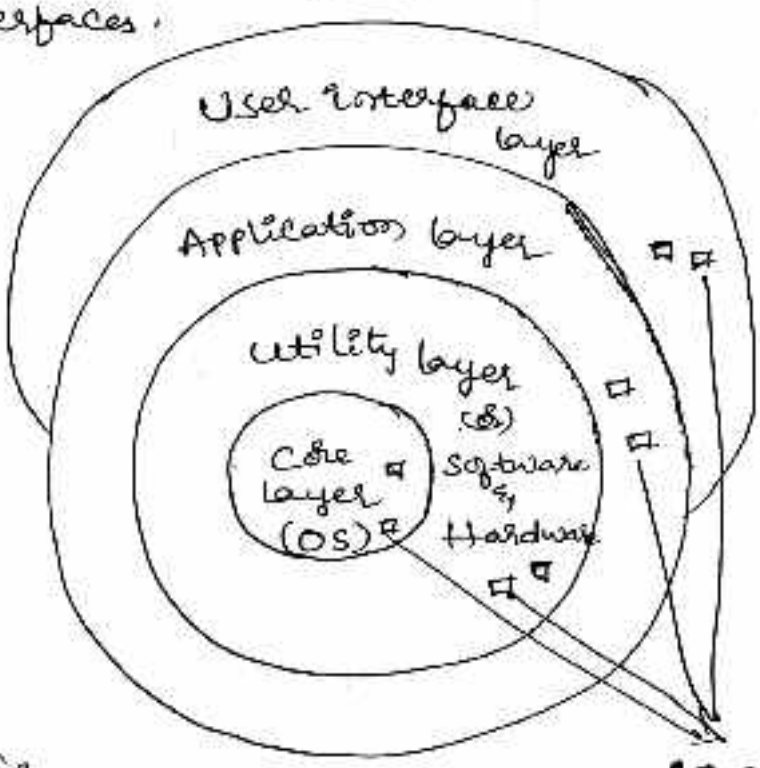


fig: layered architecture components.

The software is a collection of modules. Each module is connected to another module in order to produce outcome. A connection is the relationship between the modules.

A module that controls another module is said to be Superordinate to it and a module controlled by another module is said to be Subordinate to the controller, The arrangement is shown in figure.

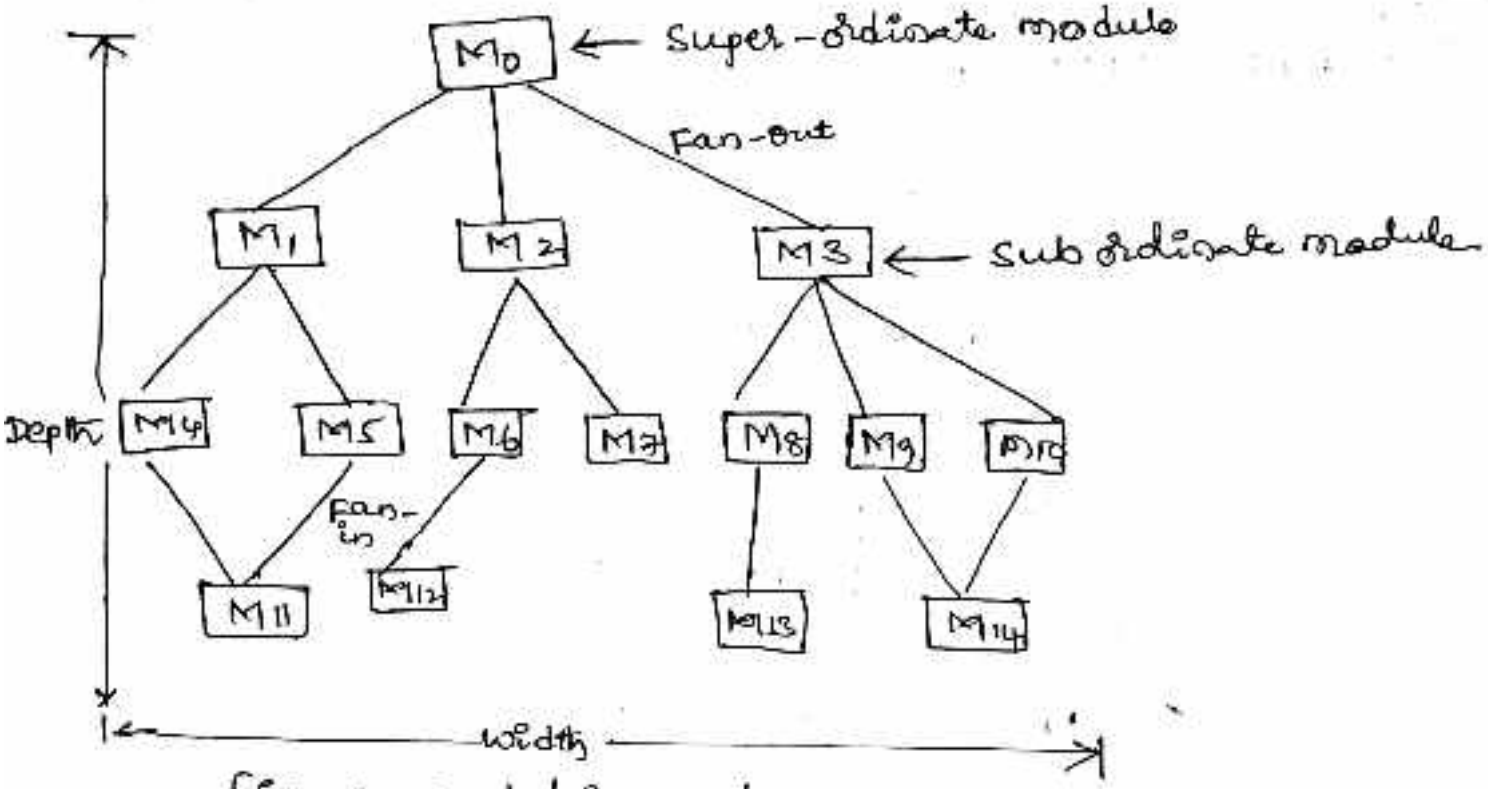


fig: control hierarchy.

Fan-out is a measure of the number of modules that are directly controlled by another module.

Fan-in indicates how many modules directly control by a given module.

- APPROACHES TO SOFTWARE DESIGN:

Design approaches are used to propose a solution for the system in a conceptual manner once all the requirements are available. A design methodology provides the techniques and guidelines for the design process of a system.

The most popular design methodologies are

1. Function-oriented design
2. Object-Oriented design.

Function-Oriented design:

→ Function Oriented design is a mature design methodology for software design.
→ The function Oriented design begins with the requirement document i.e SRS to understand the different modules.
→ During the design process, the modules are linked together to form a cohesive system.

Ex: consider a problem of admitting student in a course.
The student admission software has various sub-modules such as Online student registration, conducting Online entrance examination, Result processing, generating merit position, online counselling and course registration. All these functional requirements can be decomposed into more refined and detailed levels.
The student registration can be decomposed into online form filling, deposit registration fee, generation of admit card and attendance sheet.

Object-Oriented design:

→ Object Oriented design has become a popular design methodology in the recent years.
→ Object Oriented design deals with the real world entities of the environment for the problem solving.
→ The entities are characterized by objects. Similar objects are combined into a group called a class.
→ UML (Unified Modelling Language) diagrams are discussed as graphical representation using OOD.

Design principles

1. Abstraction → describing the problem at high level representation
2. Information hiding
   - Abstraction
   - Encapsulation
3. Functional decomposition → It is the process of partitioning a large into small
4. Design Strategy
   - Top down Strategies
   - Bottom up Strategies
5. Modularity → A module is a part of software system. It interface between modules.

Important Questions

1. Explain briefly Requirement engineering process?
2. Explain briefly SRS (Software requirement Specification) structure?
3. What are the characteristics of design?
4. Explain briefly ~~Stress~~ Requirement Analysis?
5. Explain briefly cohesion and coupling?
6. Explain briefly Overview of design process?
7. Explain briefly layered arrangement modules with neat sketch?