



Software Engineering



UNIT 3

Structured Systems Analysis and Design System Analysis

Structured System Analysis

- System Analysis is a process of understanding the system requirements and its environment.
- It is one of the initial stages of the software development life cycle.
- System analysis is the process of breaking the system down into its individual components and understanding how each component interacts with the other components.
- In this process, the analyst collects the requirements of the system and documents them.

Importance of Structured System Analysis

- Improves the user experience
- Reduce errors or inefficiencies
- Identifies potential issues in code
- Helps businesses to improve their systems

Characteristics of Structured System Analysis

- It is the study of the existing system to identify the problem areas.
- It is a process of understanding the system requirements and its environment.
- It involves gathering and understanding the user's requirements.
- It involves analyzing the system in terms of its current and future needs.

Advantages of Structured System Analysis

- It helps to identify the problems and their causes.
- It helps to understand the functional and non-functional requirements of the system.
- It helps to develop better solutions.
- It helps identify the areas of improvement.

Limitations of System Analysis

- It can be time-consuming.
- It can be costly.
- It can be difficult to get accurate information.

What is SSADM ?

- Structured System Analysis Development Methodology
- It provides a structured approach to system development by using a series of well-defined phases, tools, and techniques to ensure that a system is developed in a controlled, organized manner. SSADM is particularly focused on improving the quality of system requirements and creating detailed system models before coding begins.
- SSADM is in fact a modified form of SDLC. So it is called as SSADM or SDLC using structured techniques.

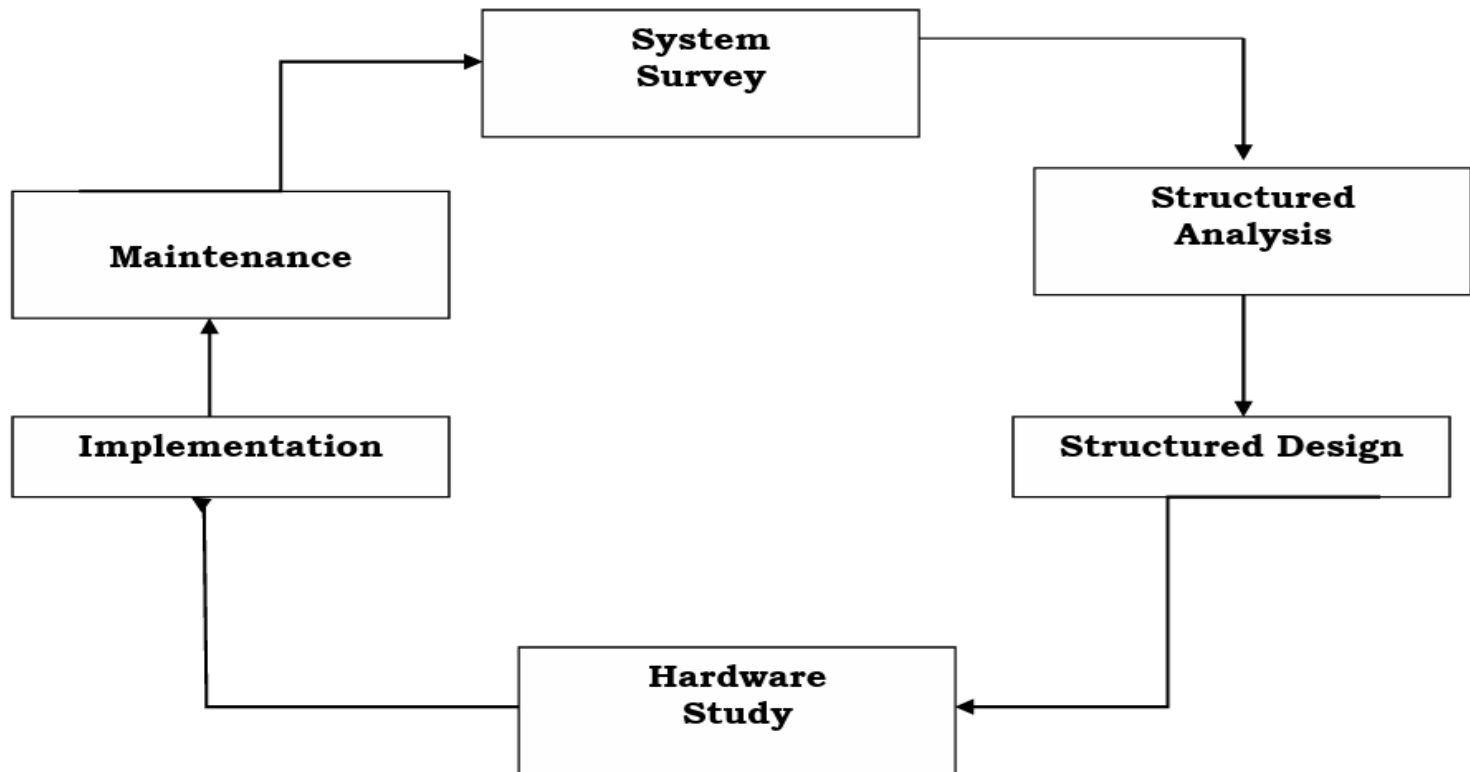
Need of SSADM or Limitations of SDLC

- Interaction with the user is limited.
- The systems analyst is constantly Overcome with the business and technical details of the system.
- In SDLC, It becomes difficult for the user to understand how the system parts fit together.
- The complete package of the system is viewed after the fully implementation of the product.
- All the system documents are prepared at the end of the project.

Structured System Analysis Development Methodology (SSADM)

- SSADM consists of:
 1. **System survey**
 2. **Structured analysis**
 3. **Structured design**
 4. **Hardware study**
 5. **Implementation**
 6. **Maintenance**

SSADM



[Figure of Structured System Analysis & Design Methodology]

Tools for Analysis

1. Decision tree
2. Decision table
3. Structured English
4. Data flow diagram
5. Entity Relationship Diagram
6. Data dictionary

Decision Tree

- A **decision tree** in software engineering (SE) is a tool used for decision-making and problem-solving. It is a flowchart-like model that helps break down complex decisions into simpler, structured choices and outcomes, providing a clear visual representation of the paths and decisions involved.

Components of a Decision Tree

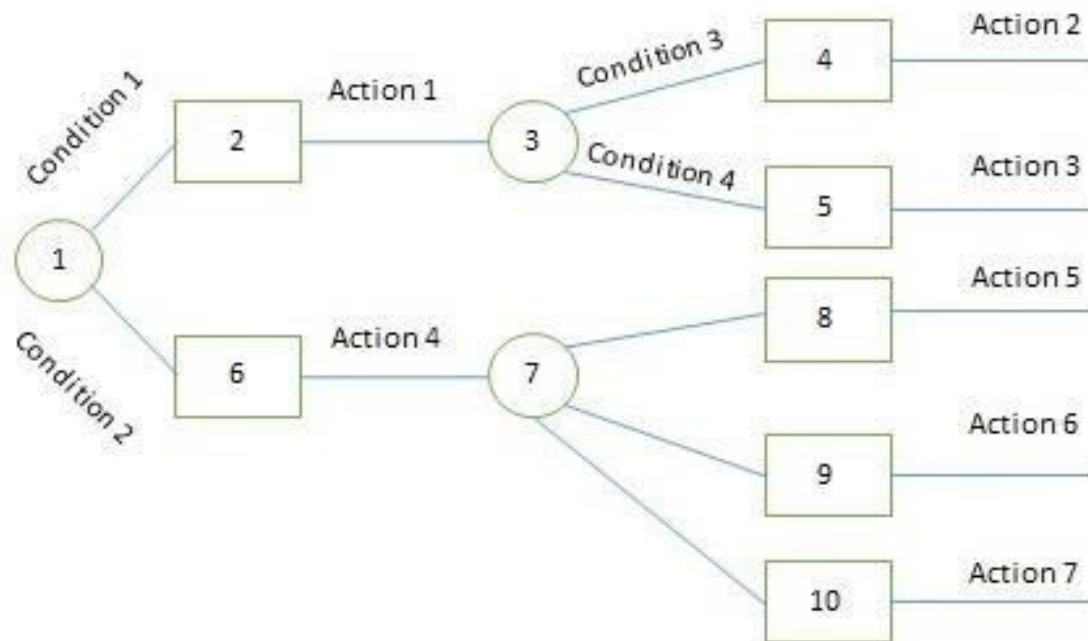
Root Node: The starting point of the decision tree, representing the initial decision or problem.

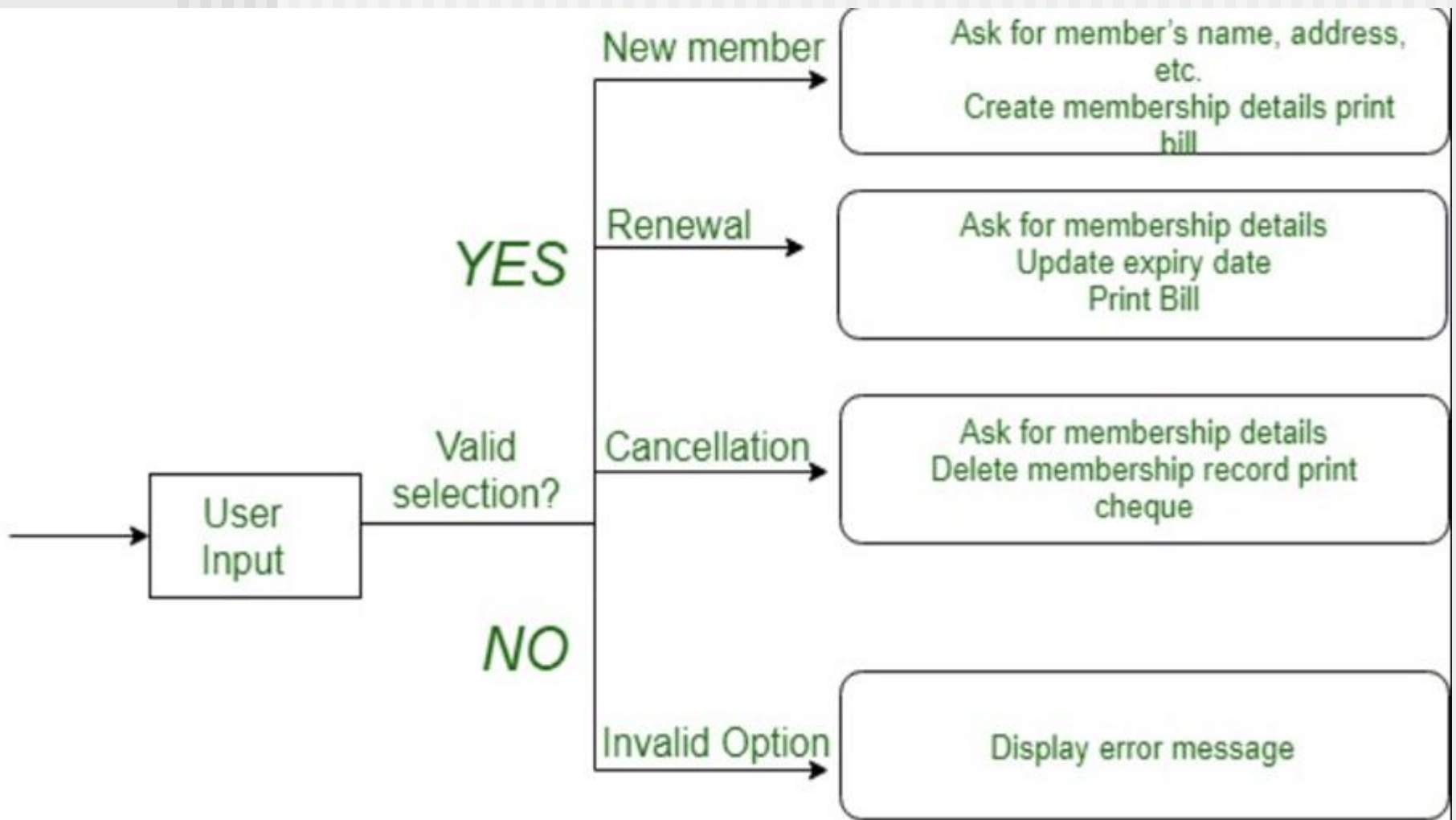
Branches: Lines that extend from a node and represent possible choices or actions based on the decision.

Decision Nodes: Points where a decision needs to be made, typically represented by squares.

Leaf Nodes: The endpoints of the tree, representing the outcome or final decision, typically represented by circles or rectangles.

Decision Tree





Decision tree for LMS

Decision Table

- Decision tables are a method of describing the complex logical relationship in a precise manner which is easily understandable.
- It is useful in situations where the resulting actions depend on the occurrence of one or several combinations of independent conditions.
- It is a matrix containing row or columns for defining a problem and the actions.

Components of a Decision Table

- **Condition Stub** – It is in the upper left quadrant which lists all the condition to be checked.
- **Action Stub** – It is in the lower left quadrant which outlines all the action to be carried out to meet such condition.
- **Condition Entry** – It is in upper right quadrant which provides answers to questions asked in condition stub quadrant.

Components of a Decision Table

- **Action Entry** – It is in lower right quadrant which indicates the appropriate action resulting from the answers to the conditions in the condition entry quadrant.
- The entries in decision table are given by Decision Rules which define the relationships between combinations of conditions and courses of action.

CONDITIONS STEP 1 STEP 2 STEP 3 STEP 4

Condition 1

Condition 2

Condition 3

Condition 4

Decision Table: Combinations

CONDITIONS STEP 1 STEP 2 STEP 3 STEP 4

Condition 1 Y Y N N

Condition 2 Y N Y N

Condition 3 Y N N Y

Condition 4 N Y Y N

Structured English

- Structure English is derived from structured programming language which gives more understandable and precise description of process.
- It is based on procedural logic that uses construction and imperative sentences designed to perform operation for action.
- It is best used when sequences and loops in a program must be considered and the problem needs sequences of actions with decisions.
- It does not have strict syntax rule.
- It expresses all logic in terms of sequential decision structures and iterations.

```
IF customer has a Bank Account THEN
  IF Customer has no dues from previous account THEN
    Allow loan facility
  ELSE
    IF Management Approval is obtained THEN
      Allow loan facility
    ELSE
      Reject
    ENDIF
  ENDIF
ELSE
  Reject
ENDIF
EXIT
```

Data Flow Diagram (DFD)

- It is a graphical tool, useful for communicating with users ,managers and other personnel, it is useful for analyzing existing as well as proposed system.
- The flow of data of a system or a process is represented by DFD.
- It also gives insight into the inputs and outputs of each entity and the process itself.
- DFD does not have control flow and no loops or decision rules are present.

Data Flow Diagram (DFD)

- Specific operations depending on the type of data can be explained by a flowchart.
- It should be pointed out that a DFD is not a flowchart.
- In drawing the DFD, the designer has to specify the major transforms in the path of the data flowing from the input to the output.
- DFDs can be hierarchically organized, which helps in progressively partitioning and analyzing large systems.

Data Flow Diagram (DFD)



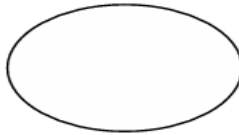
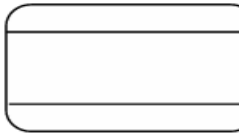




- It provides an overview of
 - What data is system processes.
 - What transformation are performed.
 - What data are stored.
 - What results are produced , etc.
- Data Flow Diagram can be represented in several ways.
- The DFD belongs to structured-analysis modeling tools.

Characteristics of DFD

- DFDs are commonly used during problem analysis.
- DFDs are quite general and are not limited to problem analysis for software requirements specification.
- DFDs are very useful in understanding a system and can be effectively used during analysis.
- It views a system as a function that transforms the inputs into desired outputs.
- The DFD aims to capture the transformations that take place within a system to the input data so that eventually the output data is produced.
- The processes are shown by named circles and data flows are represented by named arrows entering or leaving the bubbles.
- A rectangle represents a source or sink and it is a net originator or consumer of data. A source sink is typically outside the main system of study.

Components of DFD

The Data Flow Diagram has 4 components:

Sr No.	Symbol Name	Symbol	
		Yordon	Gane & Sarson
1.	External entity: a source or destination of data which is external to the system. E.g. supplier, customer etc.		
2.	Process: Here flow of data is transformed. E.g. verify credits, update inventory file.		
3.	Data flow: it is a packet of data. It may be in the form of a document, letter, telephone call etc.		
4.	Data Store: any stored data but with no reference to the physical method of storing. E.g. inventory master file, customer master file etc.		

E-R Diagram

- **ER Diagram** stands for Entity Relationship Diagram, also known as ERD is a diagram that displays the relationship of entity sets stored in a database.
- ER diagrams help to explain the logical structure of databases. ER diagrams are created based on three basic concepts: entities, attributes and relationships.
- ER Diagrams contain different symbols that use rectangles to represent entities, ovals to define attributes and diamond shapes to represent relationships.

Components of the ER Diagram

■ **Entities :**

An entity can be place, person, object, event or a concept, which stores data in the database. The characteristics of entities are must have an attribute, and a unique key. There are two types of entities:

- Strong Entity : Entity with primary key
- Weak Entity : Entity without primary key

Components of the ER Diagram

■ **Attributes :** _____

The properties that characterize an entity set are called its attributes. Attribute is also called data item, data element, data field, item, elementary item or object property. Types of Attributes:

- Single valued attributes
- Composite attributes
- Derived attributes
- Multi valued attributes
- Attributes with null values

Components of the ER Diagram

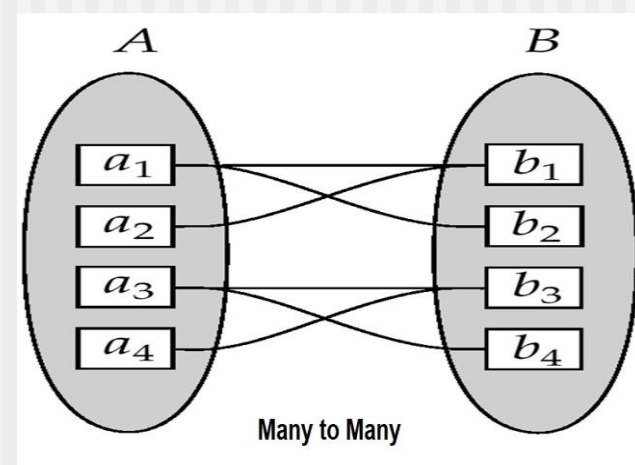
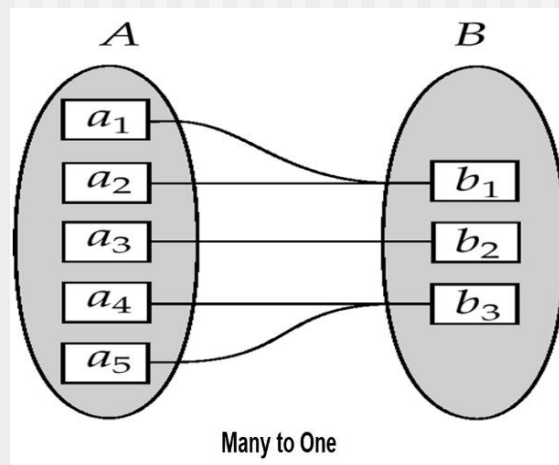
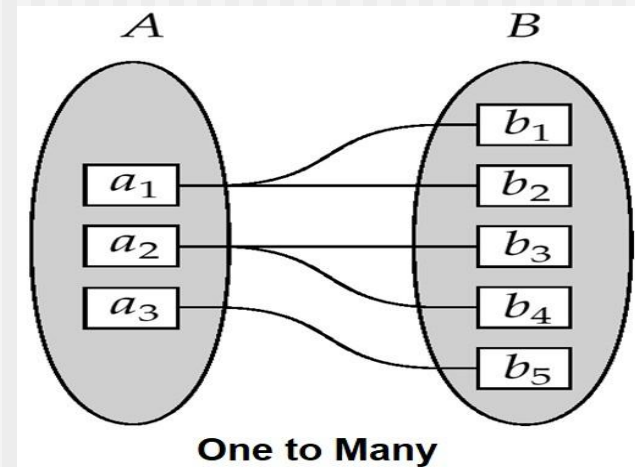
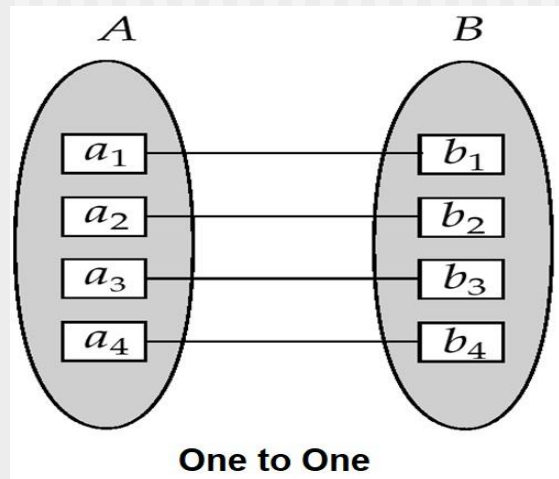
■ Relationships :

An association among entities is called relationship. A collection of relationships of the same type is called a relationship set. The relationship set is used in data modeling to represent an association between entity sets & itself.








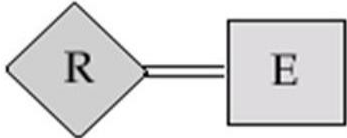


Degree of Relationship set :

- Unary Relationship
- Binary Relationship
- Ternary Relationship

Relationship Types / Mapping Cardinality:



Symbols used in E-R Diagram

	Entity Set		Attribute
	Weak Entity Set		Multivalued Attribute
	Relationship Set		Derived Attribute
	Identifying Relationship Set for Weak Entity Set		Total Participation of Entity Set in Relationship
	Primary Key		Discriminating Attribute of Weak Entity Set

Data Dictionary (DD)

- A data dictionary is a collection of data about data.
- It maintains information about the definition, structure, and use of each data element that an organization uses.
- Data dictionary contains description & definition consulting the data structure, data elements, their interrelationship & other characteristics of a system.

client_id	client_name	Password	Contact_no	email
1	Raju	12345	777 777 777	raju@email.com
2	Rohan	Abcdef	666 666 666	rohan@email.com
3	Sohan	54321	555 555 555	sohan@email.com
4	Mohan	14145	222 222 222	mohan@email.com

Field Name	Data type	Field Length	Constrains	Description
client_id	number	10	primary key	Client id, Auto generated
client_name	varchar	20	not null	Name of client
password	varchar	30	not null	Login password
contact_no	number	10	not null	Contact of client
email	varchar	40	not null	Client email

System Design

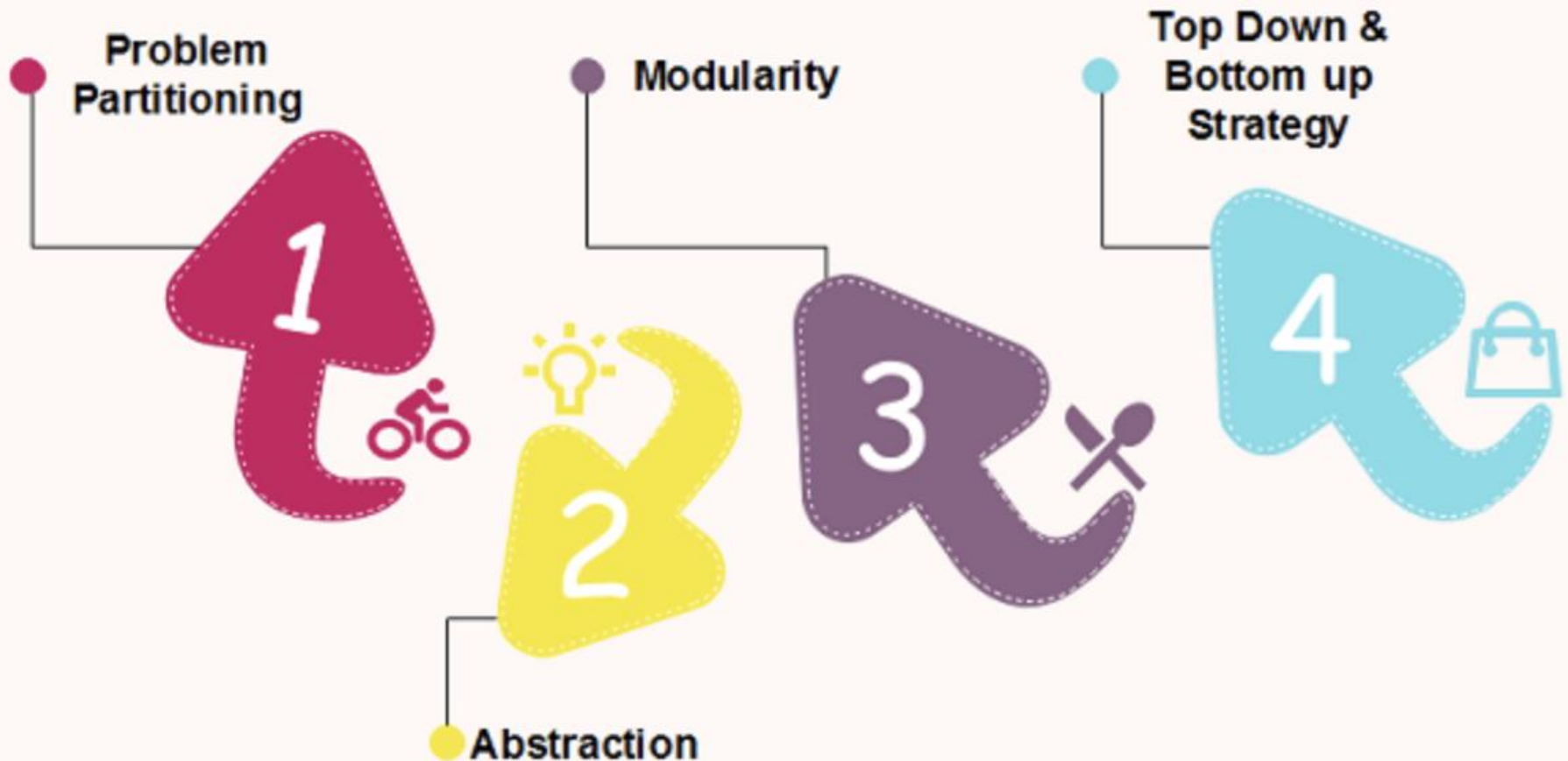
- System design is the process of defining the architecture, components, modules, interfaces, and data of a system to satisfy specific requirements. It involves making high-level decisions about how a system will function, how different parts will interact, and how it will achieve its objectives.
- The goal of system design is to create a blueprint for the development and implementation of a computer system that meets the needs of the users and stakeholders.

Design concepts

- It describes how you plan to solve the problem of designing software, and the logic, or thinking behind how you will design software. It allows the software engineer to create the model of the system software or product that is to be developed or built. The software design concept provides a supporting and essential structure or model for developing the right software.

Design concepts

Software Design Principles



Design concepts

Problem Partitioning

For small problem, we can handle the entire problem at once but for the significant problem, divide the problems and conquer the problem it means to divide the problem into smaller pieces so that each piece can be captured separately.

Abstraction

An abstraction is a tool that enables a designer to consider a component at an abstract level without bothering about the internal details of the implementation. Abstraction can be used for existing element as well as the component being designed

Design concepts

Modularity

Modularity specifies to the division of software into separate modules which are differently named and addressed and are integrated later on in to obtain the completely functional software.

Top-down Approach: This approach starts with the identification of the main components and then decomposing them into their more detailed sub-components.

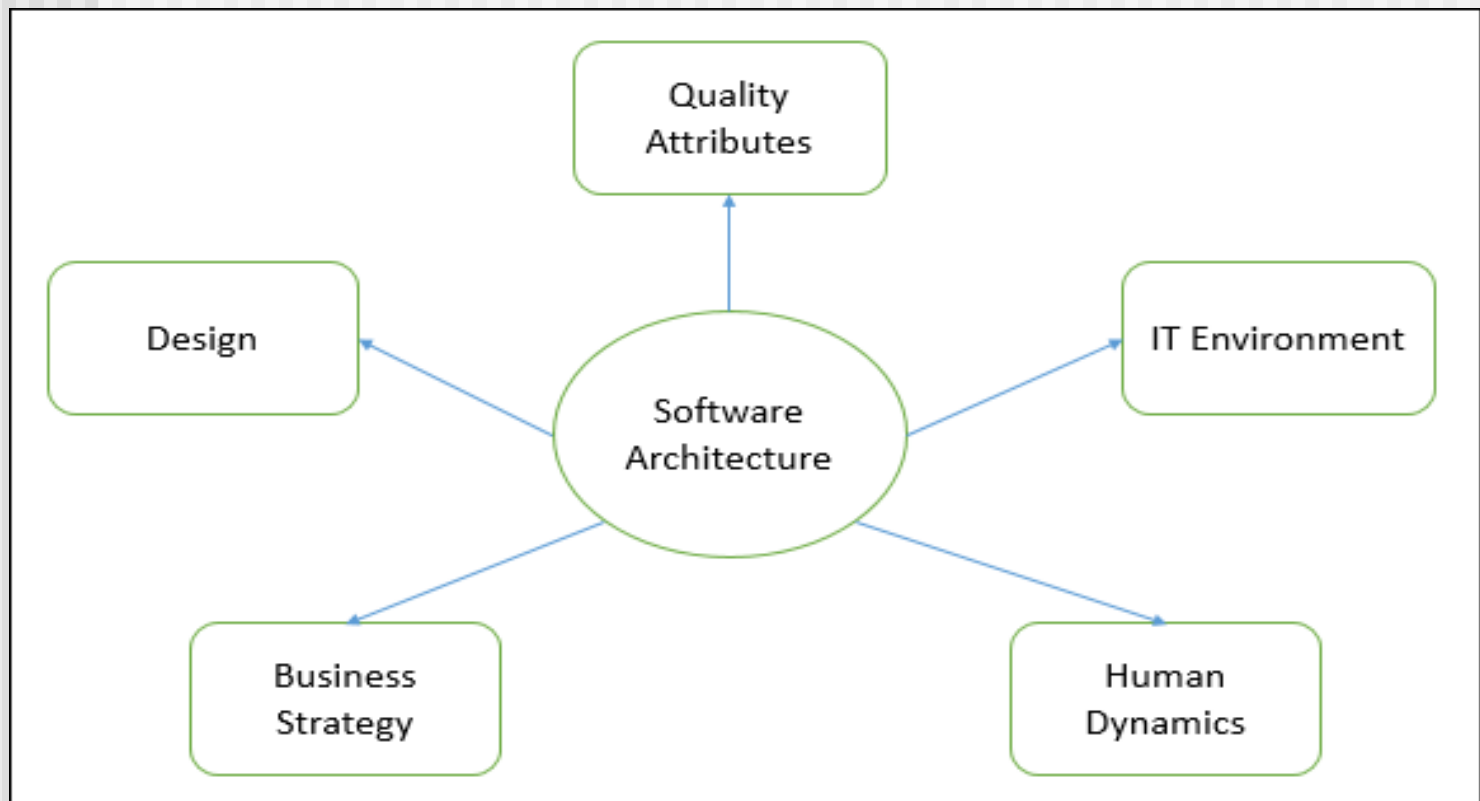
Bottom-up Approach: A bottom-up approach begins with the lower details and moves towards up the hierarchy. This approach is suitable in case of an existing system.

Design Modeling

- **Design modeling** in software engineering (SE) is the process of representing the architecture, components, interfaces, and data of a system in a visual or abstract form. It provides a blueprint for how a software system will be built, serving as a guide during development and a communication tool for stakeholders. Design models offer a structured way to visualize the system's structure and behavior before implementation begins.

Software architecture

The architecture of a system describes its major components, their relationships (structures), and how they interact with each other.



Data design

- Data designing illustrates the types of data that are stored in the system, the relationships between them and the ways that data can be grouped or organized.
- **Data design** is the first design activity, which results in less complex, modular and efficient program structure.

Architectural styles

- The architectural style shows how to organize code, or how the system will look like from very high level to show the highest level of abstraction of the system design.
- Furthermore, when building the architectural style of the system focus on layers and modules and how they are communicating with each other.

Architectural Patterns

- The architectural pattern shows how a solution can be used to solve a recurring problem.
- It reflects how a code or components interacts with each other.
- The architectural pattern describes the architectural style of our system and provides solutions for the issues in our architectural style.

Procedural design

- It is used to model programs that have an obvious flow of data from input to output.
- It represents the architecture of a program as a set of interacting processes that pass data from one to another.
- Procedural design is also called component design.
- It is completely based on process and control specifications.
- The “state transition diagram” of the requirements analysis model is also used in component design.

Object oriented design

- In the object-oriented design method, the system is viewed as a collection of objects (i.e., entities).
- The state is distributed among the objects, and each object handles its state data.
- The tasks defined for one purpose cannot refer or change data of other objects.
- Objects have their internal data which represent their state. Similar objects create a class.
- In other words, each object is a member of some class. Classes may inherit features from the superclass.