Parul® University NAAC A++ ACCREDITED UNIVERSITY

Parul® University

SOFTWARE ENGINEERING (303105253)

Alok Singh Kushwaha

1. Structured System Design:
2. Design Concepts,
3. Design Model,
4. Software Architecture,
5. Data Design,
6. Architectural Styles and Patterns,
7. Architectural Design,
8. Alternative architectural designs,
9. Modelling Component level design and its modeling,
10. Procedural Design,
11. Object Oriented Design.
12. Data Oriented Analysis & Design :
13. Difference between Data and Information,
14. E-R Diagram,
15. Dataflow Model,
16. Control Flow Model,
17. Control and Process Specification,
18. Data Dictionary

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

1.Structured System Design

Structured System Design is a methodology in **software engineering** aimed at designing a system in a structured and organized manner.

It focuses on dividing the system into smaller, manageable parts to ensure clarity, modularity, and efficiency during development.

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

2.Design Concepts

"**Design Concepts**" in software engineering refer to the fundamental principles and practices used to guide the architecture, design, and development of software systems

1. **Abstraction**
2. **Architecture**
3. **Design Patterns**
4. **Modularity**
5. **Information Hiding**
6. **Functional Independence**
7. **Refinement**
8. **Refactoring**
9. **Object-Oriented Design Concept**

Abstraction

•Simplifying complex systems by modeling only the necessary details while ignoring the irrelevant parts.

•Types of abstraction include:

- **Data abstraction** (e.g., abstract data types)
- **Control abstraction** (e.g., encapsulating algorithms in functions)

Modularity

•Dividing a system into smaller, manageable, and independent components (modules) that can be developed and tested separately.

•Promotes maintainability, scalability, and reusability.

Encapsulation(Information Hiding)

- Restricting access to the internal details of a module and exposing only the necessary functionality.
- Helps protect the integrity of the system and reduces interdependencies.

Separation of Concerns

- Dividing a software application into distinct features that overlap as little as possible (e.g., separating UI from business logic).
- Leads to cleaner and more maintainable code.

Refinement

- Gradually adding detail to a system design, starting from a high-level abstract view.
- Involves moving from general to specific in the design process.

Design Patterns

- Proven solutions to common software design problems.
- Examples: Singleton, Factory, Observer, Strategy, and MVC patterns.

SOLID Principles

- A set of five principles (SRP, OCP, LSP, ISP, DIP) that guide object-oriented design and architecture.

SRP (Single Responsibility Principle)

OCP (Open-Closed Principle)

LSP (Liskov Substitution Principle)

ISP (Interface Segregation Principle)

DIP (Dependency Inversion Principle)

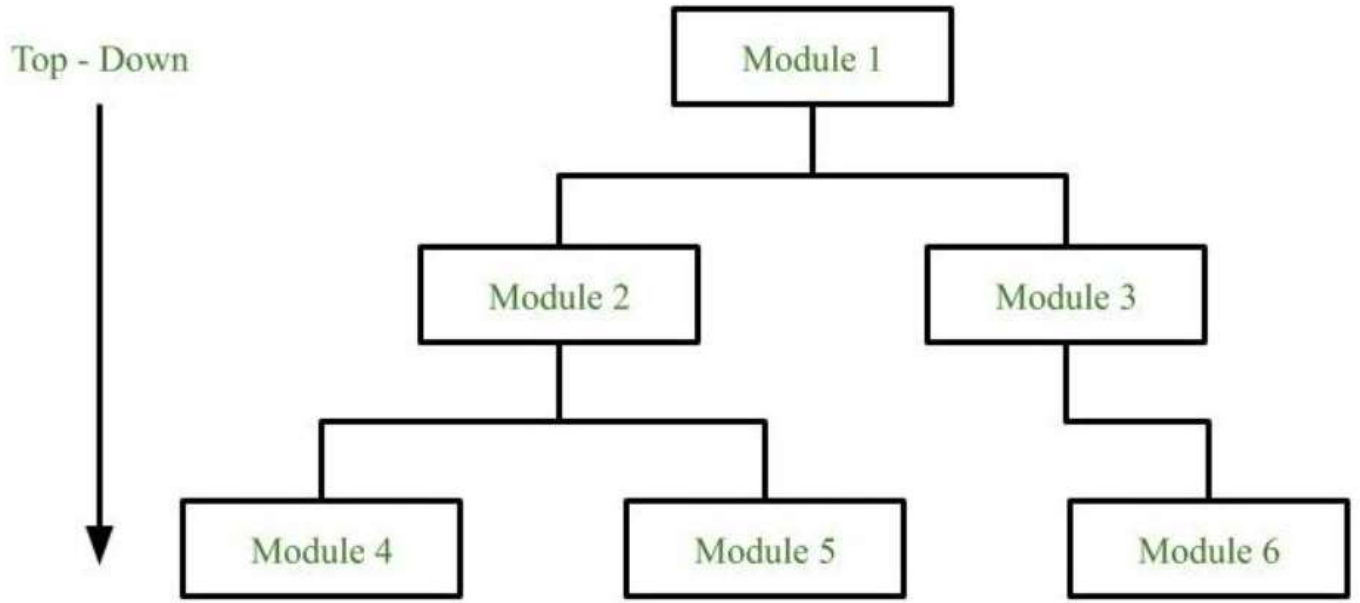
Parul® University

NAAC A++ ACCREDITED UNIVERSITY

3.Design Models

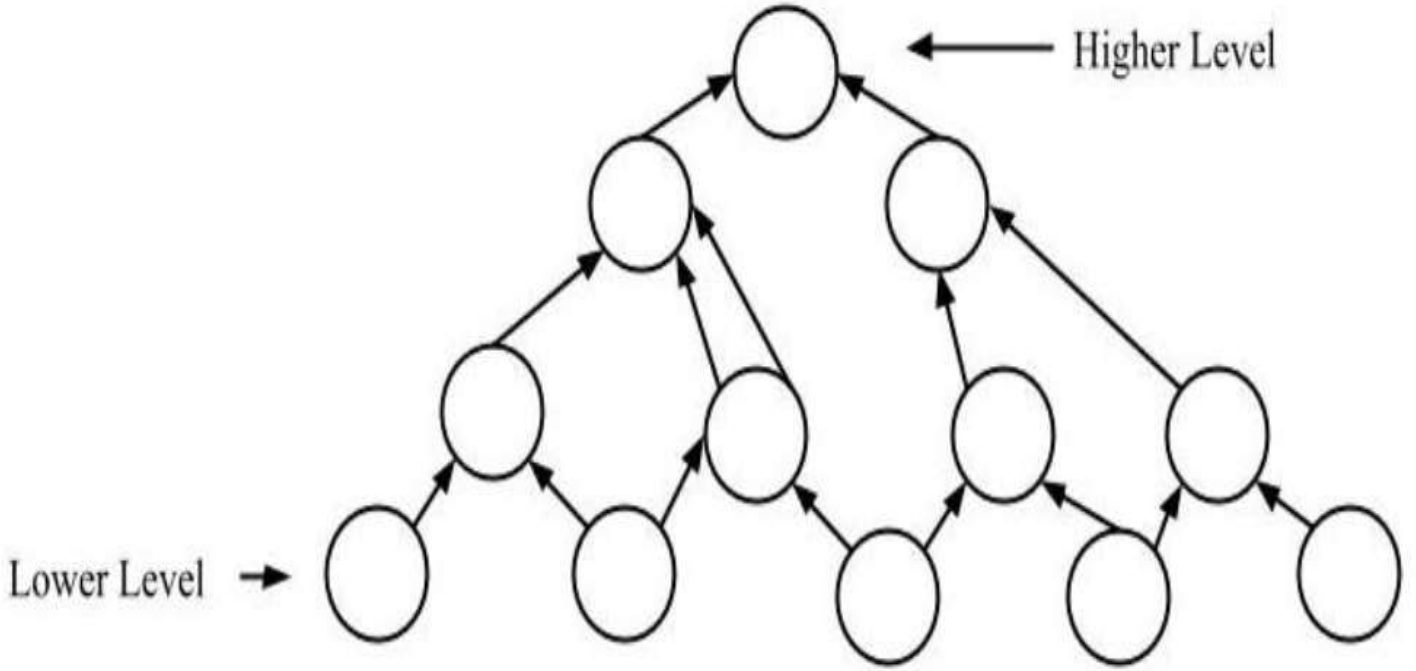
System Design Strategy refers to the approach that is taken to design a software system.

1. Top-Down Design: This strategy starts with a high-level view of the system and gradually breaks it down into smaller, more manageable components.



Top-down approach

2.Bottom-Up Design: This strategy starts with individual components and builds the system up, piece by piece.



Bottom-up approach

Advantages of Bottom-up approach:

- The economics can result when general solutions can be reused.
- It can be used to hide the low-level details of implementation and be merged with the top-down technique.

Disadvantages of Bottom-up approach:

- It is not so closely related to the structure of the problem.
- High-quality bottom-up solutions are very hard to construct.
- It leads to the proliferation of ‘potentially useful’

3.Iterative Design: This strategy involves designing and implementing the system in stages, with each stage building on the results of the previous stage.

4.Incremental Design: This strategy involves designing and implementing a small part of the system at a time, adding more functionality with each iteration.

5.Agile Design: This strategy involves a flexible, iterative approach to design, where requirements and design evolve through collaboration between self-organizing and cross-functional teams.

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

4.Software Architecture

Definition:

Software architecture refers to the high-level structure of a software system and the discipline of creating such structures.

It involves the set of decisions about the organization of a system, its components, their relationships, and how they interact to meet the software's functional and non-functional requirements.

Importance:

- Provides a blueprint for software development.
- Ensures scalability and robustness.
- Helps in managing complexity.
- Facilitates communication among stakeholders.

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

5.Data Design

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

DIGITAL LEARNING CONTENT

DATA

Data is collection of facts

Data is unorganized.

Data does not depend on information

Data isn't sufficient for decision-making

VS

INFORMATION

Information puts facts into context.

Information is organized

Information depends on data.

Information is sufficient for decision -making

Definition:

Data design refers to the process of structuring and organizing data to support efficient processing, storage, and retrieval while meeting the application's requirements.

Key Components:

- **Data Modeling:** Creation of data models like Entity-Relationship (ER) diagrams to represent data entities and their relationships.
- **Database Design:** Designing relational or NoSQL databases for optimized storage and retrieval.
- **Data Structures:** Selection of efficient structures like arrays, linked lists, and trees to process data in memory.

Importance:

- Ensures data integrity and security.
- Optimizes performance by reducing redundancy.
- Supports the application's functionality with efficient data access

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

6.Architectural Styles and Patterns

Architectural Styles and Patterns

Architectural Styles:

Architectural styles define a family of systems with shared characteristics and design principles.

Examples:

- **Layered Architecture:** Divides the system into layers like presentation, business logic, and data access.
- **Client-Server Architecture:** Separates the system into clients (requesters) and servers (providers).
- **Microservices Architecture:** Decomposes applications into loosely coupled, independently deployable services.

Design Patterns:

Design patterns are reusable solutions to common problems in software design.

Examples:

- **Model-View-Controller (MVC):** Separates user interface, business logic, and data.

- **Observer Pattern:** Allows objects to subscribe and receive updates when the state of another object changes.

- **Singleton Pattern:** Ensures a class has only one instance.

Importance of Styles and Patterns:

- Promotes reusability and consistency.
- Facilitates easier maintenance and scalability.
- Simplifies complex system designs.

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

7.Alternative architectural designs

Examples of Alternative Designs:

1.Event-Driven Architecture (EDA):

1. **Description:** The system is designed around the production, detection, and reaction to events.
2. **Use Case:** Real-time systems like IoT or stock trading platforms.
3. **Pros:** Decouples components, supports scalability.
4. **Cons:** Complexity in managing event flow and debugging.

2.Service-Oriented Architecture (SOA):

1. **Description:** Focuses on services that are reusable and can be composed into workflows.
2. **Use Case:** Enterprise-level systems with distributed services.
3. **Pros:** Reusability, loose coupling, and scalability.
4. **Cons:** Higher initial complexity and costs.

3.Peer-to-Peer (P2P):

1. **Description:** All nodes have equal roles in contributing to the system (no centralized server).
2. **Use Case:** File-sharing applications like BitTorrent.
3. **Pros:** Resilience, scalability, and fault tolerance.
4. **Cons:** Security and coordination challenges.

Serverless Architecture:

- **Description:** Applications are built and deployed without managing servers, using cloud services.

- **Use Case:** Event-driven systems, backend for mobile apps.

- **Pros:** Cost-effective, highly scalable.

- **Cons:** Vendor lock-in and limited control.

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

9.Modeling Component-Level Design

Component-level design focuses on defining how individual software components are designed and interact within the architecture.

Modeling Techniques:

1.Unified Modeling Language (UML):

1. **Class Diagrams:** Represent components (classes), attributes, operations, and relationships.
2. **Component Diagrams:** Show the physical layout of components in a system.

- **Data Flow Diagrams (DFDs):**
- Models the flow of data between components.
- Useful for understanding dependencies and interaction.

- **Sequence Diagrams:**
- Represents the flow of messages and events between components in a time-ordered manner.

- **State Diagrams:**
- Depict the state transitions of individual components based on events.

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

10.Procedural Design

Procedural design is a method of breaking down a system into well-defined procedures or functions.
It emphasizes a step-by-step sequence of operations to solve a problem or perform a task.

Steps in Procedural Design:

1.Define High-Level Procedures:

1. Identify the main functions required for the system.
2. Example: For a banking app, high-level procedures may include login, account management, and transactions.

- **Decompose into Sub-Procedures:**

- Break each high-level procedure into smaller, manageable tasks.
- Example: The "Transaction" procedure can be broken into "validate account," "process transaction," and "log activity.“

- **Design Control Flow:**
- Use flowcharts or pseudocode to represent the sequence and flow of operations.

1.Modularize the Design:

1. Group related procedures into modules for reusability.
2. Example: A "User Management" module may include login, logout, and password reset functions.

Advantages of Procedural Design:

- Simple and easy to understand.
- Facilitates reuse of code.
- Clear flow of logic and operations.

Modeling Techniques:

1.Flowcharts:

1. Visualize the steps and decision points in a procedure.

2.Pseudocode:

1. Descriptive representation of the algorithm or logic in plain language.

3.Hierarchy Charts:

1. Depict the relationships between procedures and sub-procedures.

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

11.Object Oriented Design.

Object-Oriented Design (OOD)

Definition:

Object-Oriented Design is a software design methodology that uses objects as the fundamental building blocks of the system.

It focuses on defining the system in terms of classes, objects, their attributes, behaviors (methods), and the relationships among them.

Key Concepts in OOD:

1.Objects:

1. Real-world entities or conceptual entities represented in software.
2. Example: A "Car" object with attributes like color, brand, and methods like drive() and stop().

2.Classes:

1. Blueprints for objects that define their properties and behaviors.
2. Example: A "Person" class might define attributes like name and age and methods like walk() and speak().

•**Encapsulation:** Bundling data (attributes) and behaviors (methods) into

• a single unit and restricting direct access to the data.

•Example: Private variables with public getter and setter methods.

•**Inheritance:**

•Mechanism to create new classes based on existing ones, promoting code reuse.

•Example: A "Dog" class can inherit attributes and methods from an "Animal" class.

1.Polymorphism:

1. The ability for different objects to be accessed through the same interface.
2. Example: A method fly() may behave differently for "Bird" and "Airplane."

2.Abstraction:

1. Focusing on essential features and hiding the implementation details.
2. Example: A "Shape" class with a draw() method that is implemented differently in subclasses like "Circle" and "Rectangle."

Process of OOD:

1.Identify Objects and Classes:

1. Extract objects from the problem domain.
2. Example: For an e-commerce application, objects could be Customer, Product, and Order.

2.Define Relationships:

1. Specify associations, dependencies, and aggregations among objects.
2. Example: A Customer "places" an Order.

1.Design Class Diagrams:

1. Use UML to create class diagrams showing classes, attributes, methods, and relationships.

2.Define Methods:

1. Identify and define the methods that encapsulate the behavior of objects.

Advantages:

- Promotes modularity and code reuse.
- Easier to model real-world scenarios.
- Enhances maintainability and scalability.

Disadvantages:

- Initial design can be complex and time-consuming.
- Overhead in terms of system performance (e.g., in highly abstract designs).

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

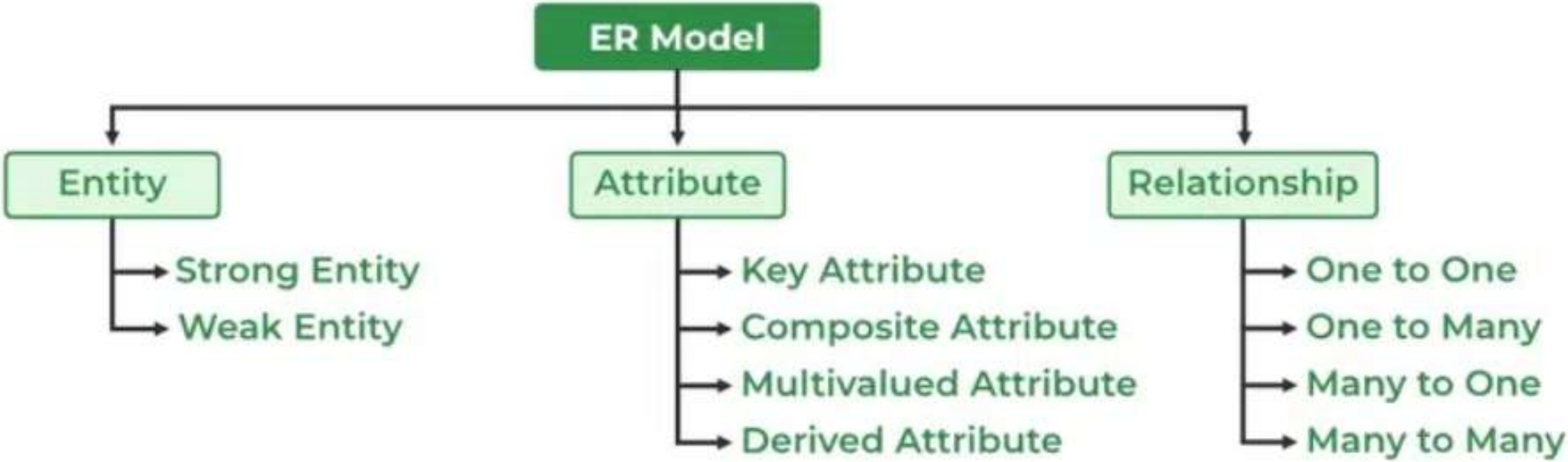
13.Difference between Data and Information

Key Differences

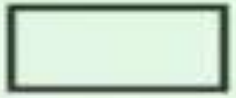
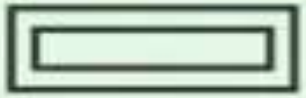
Aspect	Data	Information
Definition	Raw, unprocessed facts or figures.	Processed data with meaning and context.
Meaning	Lacks context and significance.	Has context and significance.
Form	Numbers, symbols, raw observations.	Structured, meaningful sentences or results.
Dependency	Independent; does not rely on information.	Derived from data after processing.
Example	101, 42.5, ABC.	"User 101 is John," "Temperature: 42.5°C."
Purpose	Collected for analysis or further use.	Provides insights for decision-making.

14.E-R Diagram

An **Entity-Relationship Diagram (ERD)** is a conceptual representation of data in a system, focusing on the entities, their attributes, and the relationships among them.



Components of ER Diagram

Figures	Symbols	Represents
Rectangle		Entities in ER Model
Ellipse		Attributes in ER Model
Diamond		Relationships among Entities
Line		Attributes to Entities and Entity Sets with Other Relationship Types
Double Ellipse		Multi-Valued Attributes
Double Rectangle		Weak Entity

1.Entities: Objects or concepts within the system.

- **Strong Entities**: Independent objects (e.g., Customer).
- **Weak Entities**: Dependent on another entity (e.g., OrderItem).

2.Attributes: Characteristics of entities or relationships.

- **Key Attributes**: Unique identifier for an entity (e.g.,Customer_ID).

Derived Attributes: Calculated attributes (e.g., Age from Date_of_Birth).

3.**Relationships:** Associations between entities.Cardinality: Defines the number of relationships (e.g., 1:1, 1:N, M:N).

Participation: Total (mandatory) or partial (optional).

15.Dataflow Model

The **Dataflow Model** is a conceptual representation used in software engineering, systems design, and analysis to illustrate how data moves through a system.

It focuses on the flow of data between various components, processes, and storage elements, offering a high-level overview of a system's operation.

The **Dataflow Model** represents how data moves through a system or a process.

It's widely used in computational models, software design, and hardware systems to describe systems where the output is determined by the flow of data between operations.

Data-flow diagrams (DFDs) model a perspective of the system that is most readily understood by users – the flow of information through the system and the activities that process this information.

Example :Online ordering system:

1. **Processes**: Validate Order, Calculate Total, Send Confirmation Email.
2. **Data Flows**: Customer input → Validate Order → Database → Calculate Total.
3. **Data Stores**: User Information Database, Orders Database.
4. **External Entities**: Customer, Payment Gateway.

Parul® University

NAAC A++ ACCREDITED UNIVERSITY

16.Control Flow Model

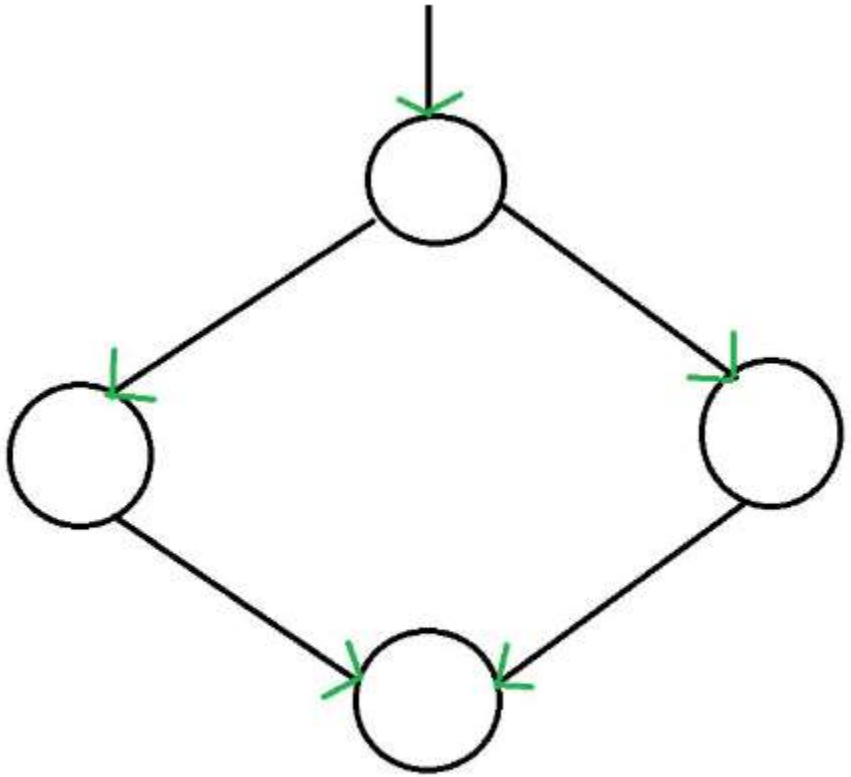
Characteristics of Control Flow Graph

The control flow graph is process-oriented.

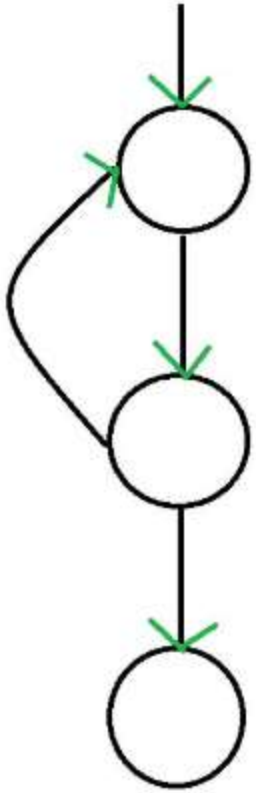
The control flow graph shows all the paths that can be traversed during a program execution.

A control flow graph is a directed graph.

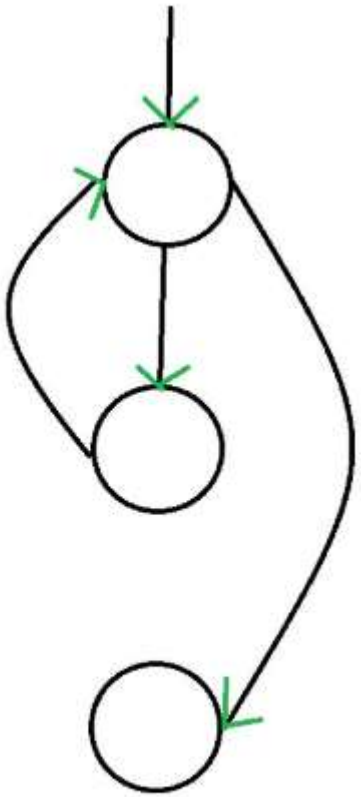
Edges in CFG portray control flow paths and the nodes in CFG portray basic blocks.



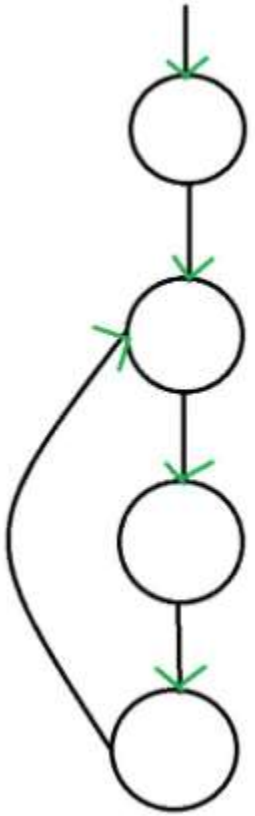
If-then-else



do-while



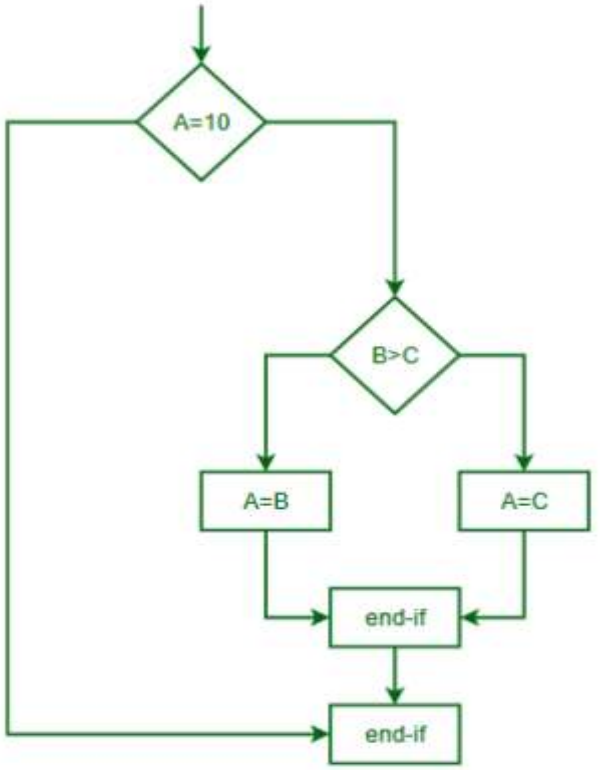
while



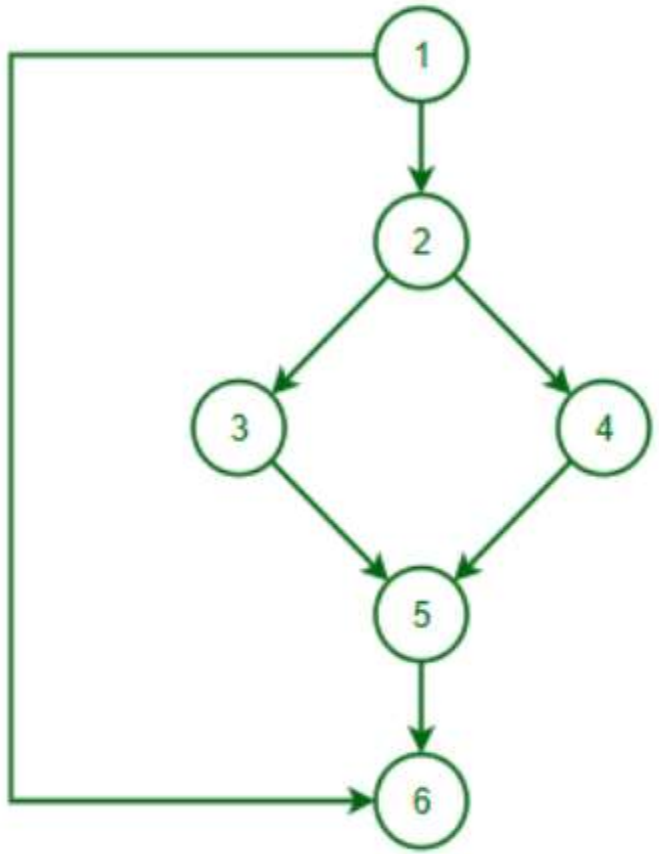
for

```
if  A = 10 then
  if B > C
    A = B
  else A = C
  endif
endif
print A, B, C
```

Flowchart

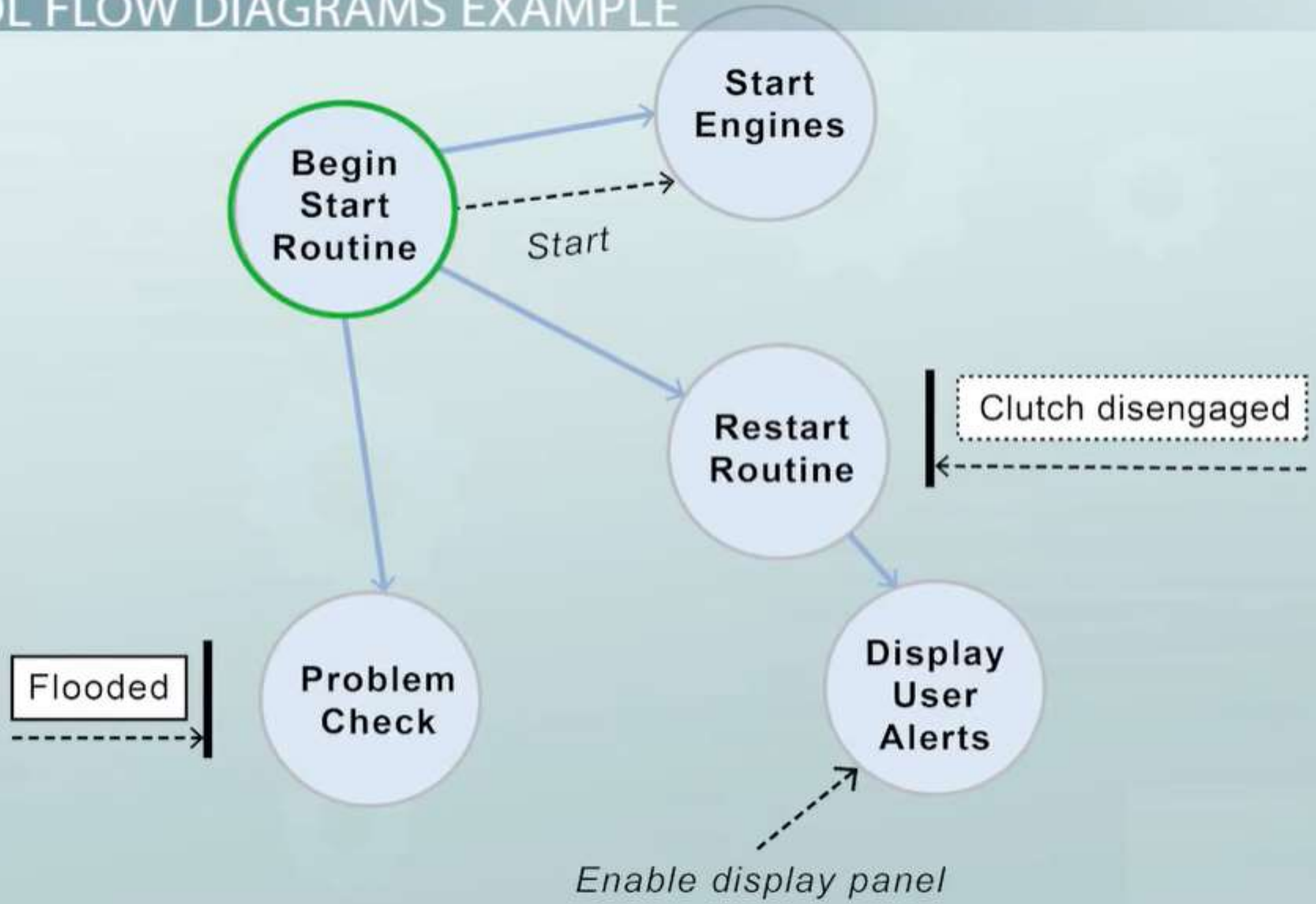


control flow graph



control flow graph

CONTROL FLOW DIAGRAMS EXAMPLE



Feature	Dataflow Model	Control Flow Model
Focus	Flow of data	Sequence of control decisions
Dependency	Data availability	Control logic (conditions, loops)
Execution	Parallel (when possible)	Sequential or branch-based
Representation	Directed acyclic graphs	Flowcharts, control flow graphs
Example Applications	Signal processing, big data	Program flow, workflows

18.Data Dictionary

Data Dictionary

A **Data Dictionary** can be defined as a collection of information on all data elements or contents of databases such as data types, and text descriptions of the system.

It makes it easier for users and analysts to use data as well as understand and have common knowledge about inputs, outputs, components of a database, and intermediate calculations.

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversity.ac.in