


6. Why modular decomposition is used in architectural design? Explain.

 **hide solution**
asked in 2069

Solution

Modular decomposition is a process of decomposing subsystems into modules. After decomposition of the system into subsystems, subsystems must be decomposed into modules.

Two modular decomposition models covered:

- An **object model** where the system is decomposed into interacting objects
- A **data-flow model** where the system is decomposed into functional modules which transform inputs to outputs. Also known as the pipeline model.

Object model

- Structure the system into a set of loosely coupled objects with well-defined interfaces.
- Object-oriented decomposition is concerned with identifying object classes, their attributes and operations.
- When implemented, objects are created from these classes and some control model used to coordinate object operations.

Advantages:

- Since objects are loosely coupled, the implementation of objects can be modified without affecting other objects.
- Objects are often representations of real-world entities so the structure of the system is readily understandable.
- As real-world entities are used in different systems, objects can also be reused.

7. What are the advantages of designing and documenting software architecture? What is repository model?

 **hide solution**
asked in 2071

Solution

The advantages of designing and documenting software architecture:

1. **Stakeholder communication:** Architecture may be used as a focus of discussion by system stakeholders.
2. **System analysis:** Well-documented architecture enables the analysis of whether the system can meet its non-functional requirements.

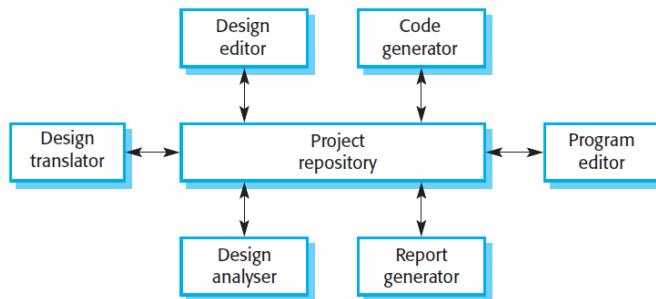
3. **Large-scale reuse:** The architecture may be reusable across a range of systems or entire lines of products.

Repository Model

A repository model is a system that will allow interfacing sub-systems to share the same data. Sub-system must exchange data so that they can work together effectively. This may be done in two ways:

1. All shared data is held in a central database that can be accessed by all subsystems. It is called repository model.
2. Each sub-system maintains its own database. Data is interchanged with other sub-systems by passing messages to them.

Example: CASE Toolset




Advantages:

- It is an efficient way to share large amounts of data. There is no need to transmit data explicitly from one sub-system to another.
- Sub-systems that produce data need not be concerned with how that data is used by other subsystems.
- Activities such as backup, security, access control and recovery from error are centralized.

Disadvantages:

- It is a compromise between the specific needs of each tool. Performance may be adversely affected by this compromise.
- Evolution may be difficult as a large volume of information is generated according to an agreed data model.
- Different sub-systems may have different requirements for security, recovery and backup policies. The repository model forces the same policy on all subsystems.

7. Discuss different activities of architectural design along with the repository model.

 **hide solution**
asked in 2072

Solution

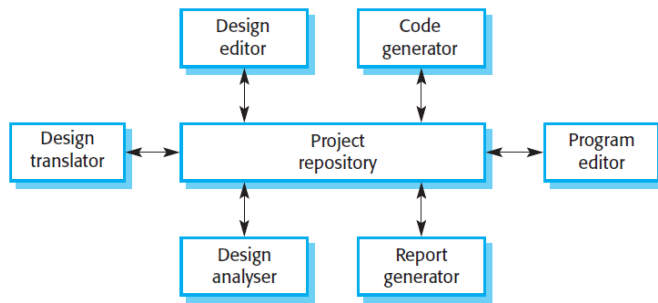
Architectural design is a process for identifying the sub-systems making up a system and the framework for sub-system control and communication. The output of this design process is a description of the software architecture. Architectural design is an early stage of the system design process. It represents the link between specification and design processes and is often carried out in parallel with some specification activities. It involves identifying major system components and their communications.

Repository Model

A repository model is a system that will allow interfacing sub-systems to share the same data. Sub-system must exchange data so that they can work together effectively. This may be done in two ways:

1. All shared data is held in a central database that can be accessed by all subsystems. It is called repository model.
2. Each sub-system maintains its own database. Data is interchanged with other sub-systems by passing messages to them.

Example: CASE Toolset




Advantages:

- It is an efficient way to share large amounts of data. There is no need to transmit data explicitly from one sub-system to another.
- Sub-systems that produce data need not be concerned with how that data is used by other subsystems.
- Activities such as backup, security, access control and recovery from error are centralized.

Disadvantages:

- It is a compromise between the specific needs of each tool. Performance may be adversely affected by this compromise.
- Evolution may be difficult as a large volume of information is generated according to an agreed data model.
- Different sub-systems may have different requirements for security, recovery and backup policies. The repository model forces the same policy on all subsystems.

7. What is client server model? Explain the advantages and disadvantages of client server model.

 **hide solution**
asked in 2074

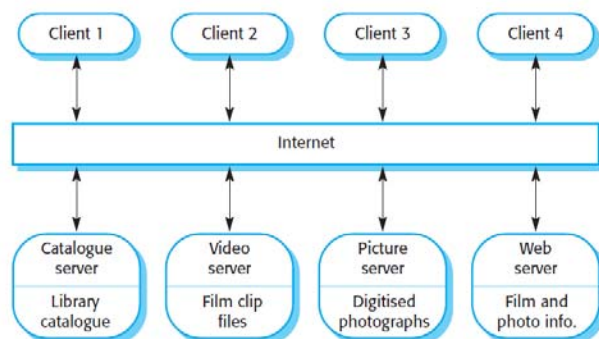
Solution

Client server model is a system model where the system is organized as a set of services and associated servers and clients that access and use the services.

The major components of this model are:

1. A **set of servers** that offer services to other sub-systems. Examples of servers are print servers that offer printing services, file servers that offer file management services and a compile server, which offers programming language compilation services.
2. A **set of clients** that call on the services offered by servers. There may be several instances of a client program executing concurrently.
3. A **network** that allows the clients to access these services.

Clients may have to know the names of the available servers and the services that they provide. However, servers need not know either the identity of clients or how many clients there are. Clients access the services provided by a server through remote procedure calls using a request-reply protocol.



Advantages:


- Distribution of data is straightforward.
- Makes effective use of networked system.
- May require cheaper hardware.
- Easy to add new servers or upgrade existing servers.

Disadvantages:

- No shared data model so sub-system use different data organization. Data interchange may be inefficient.

- Redundant management in each server.
- No central register of names and services - it may be hard to find out what servers and services are available.

7. What are the activities of architectural design process? Discuss abstract machine model.

 **hide solution**
asked in 2076

Solution

Architectural design is a process for identifying the sub-systems making up a system and the framework for sub-system control and communication. The output of this design process is a description of the software architecture. It represents the link between specification and design processes and is often carried out in parallel with some specification activities. The architectural design process is concerned with establishing a basic structural framework that identifies the major components of a system and the communications between these components.

Architectural design process includes following activities:

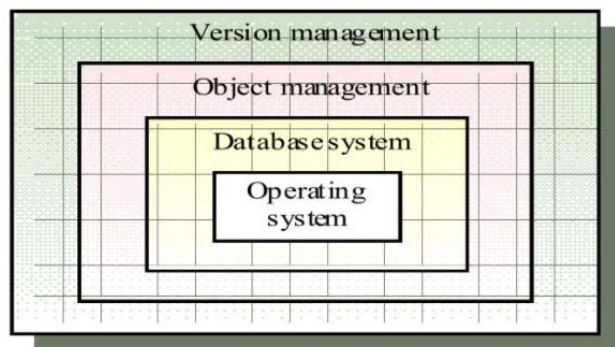
- **System structuring:** The system is decomposed into major sub-systems and communication (e.g. data sharing) mechanisms are identified.
- **Control modelling:** A model of the control relationships between the sub-systems is established.
- **Modular decomposition:** The identified sub-systems are decomposed into lower-level modules (components, objects etc.).

Abstract Machine Model

Abstract machine model is used to model the interfacing of sub-systems. It organizes the system into layers with related functionality associated with each layer. A layer provides services to the layer above it so the lowest-level layers represent core services that are likely to be used throughout the system. This model supports the incremental development of sub-systems in different layers. When a layer interface changes, only the adjacent layer is affected.

It is used when building new facilities on top of existing systems; when the development is spread across several teams with each team responsibility for a layer of functionality; when there is a requirement for multi-level security.

Example: Simple abstract machine based version management system




Advantages:

- Supports incremental development.
- Allows replacement of entire layers so long as the interface is maintained.
- Redundant facilities (e.g., authentication) can be provided in each layer to increase the dependability of the system.

Disadvantages:

- In practice, providing a clean separation between layers is often difficult and a high-level layer may have to interact directly with lower-level layers rather than through the layer immediately below it.
- Performance can be a problem because of multiple levels of interpretation of a service request as it is processed at each layer.

8. Discuss the use of control models. Differentiate between centralized control and event based control.

 **hide solution**
asked in 2071

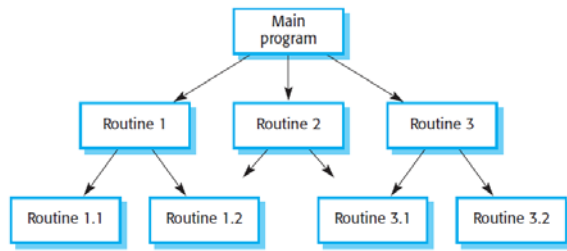
Solution

Control models are models deployed in software engineering that are concerned with the control flow between the subsystems. They are distinct from the system decomposition model.

Centralized Control Model

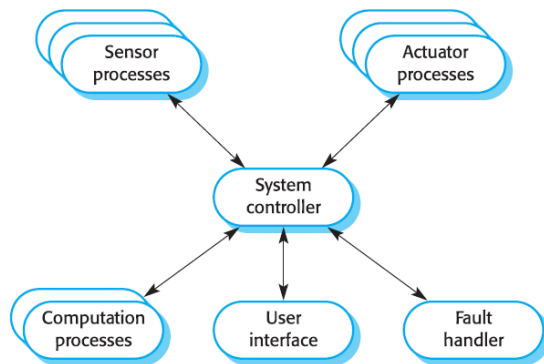
Centralized model is a formulation of centralized control in which one subsystem has overall responsibility for control and starts and stops other subsystems. It is a control subsystem that takes responsibility for managing the execution of other subsystems. Centralized models are classified into call-return and manager model.

1. Call-return Model: In call-return model, it is a model which has top-down subroutine architecture where control starts at the top of a subroutine hierarchy and moves downwards.



The call-return model is illustrated in above figure. The main program can call Routines 1, 2 and 3; Routine 1 can call Routines 1.1 or 1.2; Routine 3 can call Routines 3.1 or 3.2; and so on.

2. Manager Model: Manager model is applicable to concurrent systems. One system component controls the stopping, starting and coordination of other system processes.

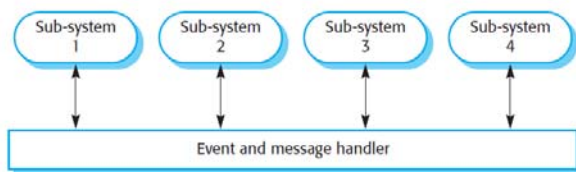


Event-based Control Model

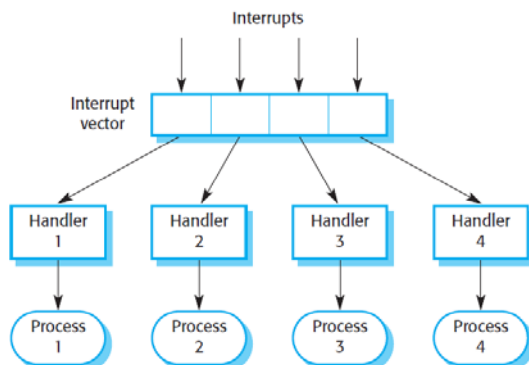
Event-based models are those in which each sub-system can respond to externally generated events from other subsystems or the system's environment. It is a system driven by externally generated events where the timing of the events is out with the control of the subsystems which process the event.

Event-based models are classified into broadcast and interrupt-driven models.

1. Broadcast models: In these models, an event is, in principle, broadcast to all sub-systems. Any sub-system, which is designed to handle that event, responds to it.



2. Interrupt-driven models: These are exclusively used in real-time systems where an interrupt handler detects external interrupts. They are then passed to some other component for processing.

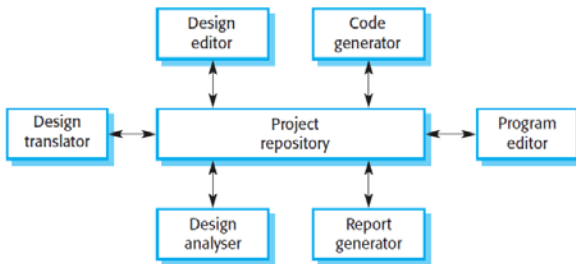


Solution

A repository model is a system that will allow interfacing sub-systems to share the same data. Sub-system must exchange data so that they can work together effectively. This may be done in two ways:

1. All shared data is held in a central database that can be accessed by all subsystems. It is called repository model.
2. Each sub-system maintains its own database. Data is interchanged with other sub-systems by passing messages to them.

Example: CASE Toolset

**Advantages:**

- It is an efficient way to share large amounts of data. There is no need to transmit data explicitly from one sub-system to another.
- Sub-systems that produce data need not be concerned with how that data is used by other subsystems.
- Activities such as backup, security, access control and recovery from error are centralized.

Disadvantages:

- It is a compromise between the specific needs of each tool. Performance may be adversely affected by this compromise.
- Evolution may be difficult as a large volume of information is generated according to an agreed data model.
- Different sub-systems may have different requirements for security, recovery and backup policies. The repository model forces the same policy on all subsystems.

8. What are control models? Differentiate between centralized control and event-based control.

hide solution
asked in 2072

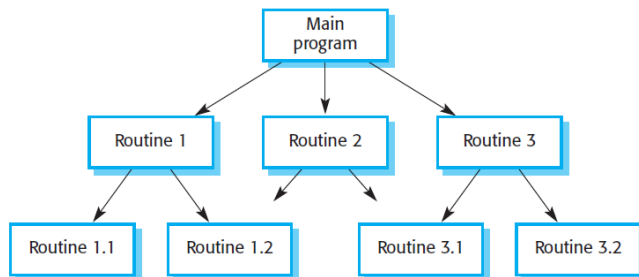
Solution

Control models are models deployed in software engineering that are concerned with the control flow between the subsystems. They are distinct from the system decomposition model.

Centralized Control Model

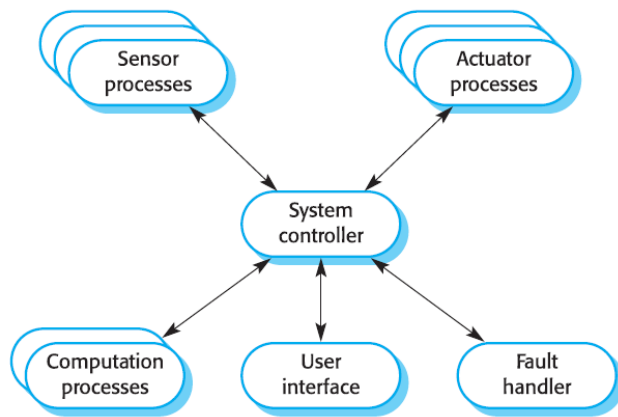
Centralized model is a formulation of centralized control in which one subsystem has overall responsibility for control and starts and stops other subsystems. It is a control subsystem that takes responsibility for managing the execution of other subsystems. Centralized models are classified into call-return and manager model.

1. Call-return Model: In call-return model, it is a model which has top-down subroutine architecture where control starts at the top of a subroutine hierarchy and moves downwards.



The call-return model is illustrated in above figure. The main program can call Routines 1, 2 and 3; Routine 1 can call Routines 1.1 or 1.2; Routine 3 can call Routines 3.1 or 3.2; and so on.

2. Manager Model: Manager model is applicable to concurrent systems. One system component controls the stopping, starting and coordination of other system processes.

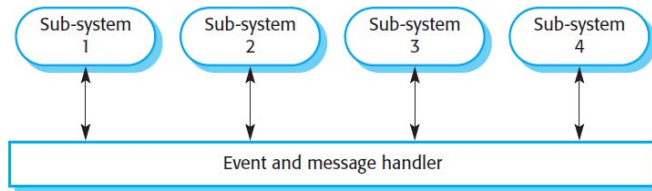


Event-based Control Model

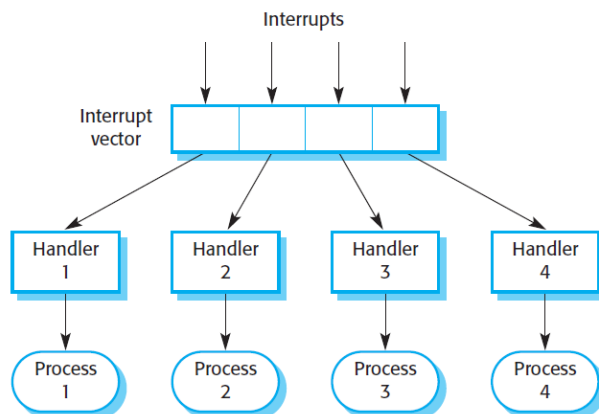
Event-based models are those in which each sub-system can respond to externally generated events from other subsystems or the system's environment. It is a system driven by externally generated events where the timing of the events is out with the control of the subsystems which process the event.

Event-based models are classified into broadcast and interrupt-driven models.


1. Broadcast models: In these models, an event is, in principle, broadcast to all sub-systems. Any sub-system, which is designed to handle that event, responds to it.



2. Interrupt-driven models: These are exclusively used in real-time systems where an interrupt handler detects external interrupts. They are then passed to some other component for processing.



8. What is modular decomposition? Discuss object oriented model of decomposition.

 **hide solution**
asked in 2076

Solution

Modular decomposition is a process of decomposing subsystems into modules. After decomposition of the system into subsystems, subsystems must be decomposed into modules.

In object oriented model of decomposition, system is decomposed into a set of communicating objects. An object-oriented, architectural model structures the system into a set of loosely coupled objects with well defined interfaces. Objects call on the services offered by other objects.

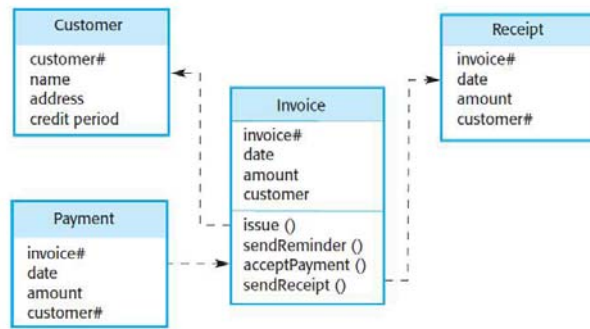


Figure: An object model of an invoice processing system

- This system can issue invoices to customers, receive payments, and issue receipts for these payments and reminders for unpaid invoices.
- Operations, if any, are defined in the lower part of the rectangle representing the object. Dashed arrows indicate that an object uses the attributes or services provided by another object.
- An object-oriented decomposition is concerned with object classes, their attributes and their operations.
- When implemented, objects are created from these classes and some control model is used to coordinate object operations.
- The Invoice class has various associated operations that implement the system functionality.

9. Explain the control models and its types.

hide solution
asked in 2068

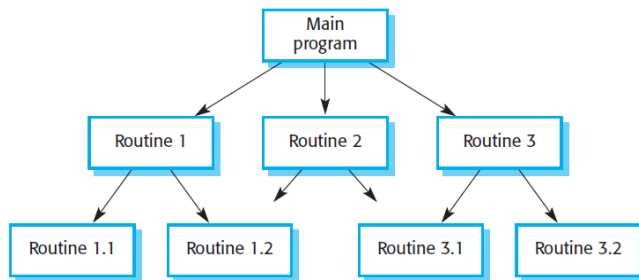
Solution

Control models are models deployed in software engineering that are concerned with the control flow between the subsystems. They are distinct from the system decomposition model. There are two types of control models: centralized and event-based control model.

Centralized Control Model

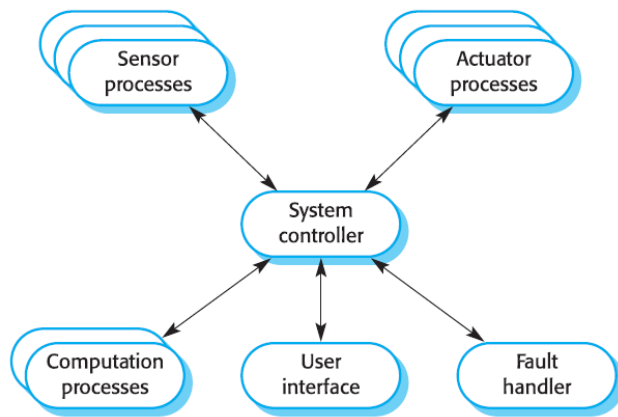
Centralized model is a formulation of centralized control in which one subsystem has overall responsibility for control and starts and stops other subsystems. It is a control subsystem that takes responsibility for managing the execution of other subsystems. Centralized models are classified into call-return and manager model.

1. Call-return Model: In call-return model, it is a model which has top-down subroutine architecture where control starts at the top of a subroutine hierarchy and moves downwards.



The call-return model is illustrated in above figure. The main program can call Routines 1, 2 and 3; Routine 1 can call Routines 1.1 or 1.2; Routine 3 can call Routines 3.1 or 3.2; and so on.

2. Manager Model: Manager model is applicable to concurrent systems. One system component controls the stopping, starting and coordination of other system processes.

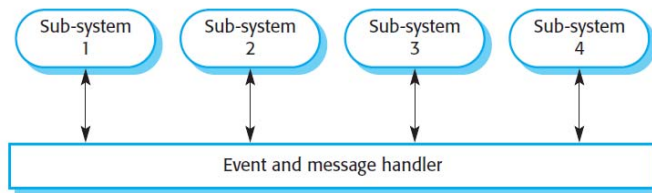


Event-based Control Model

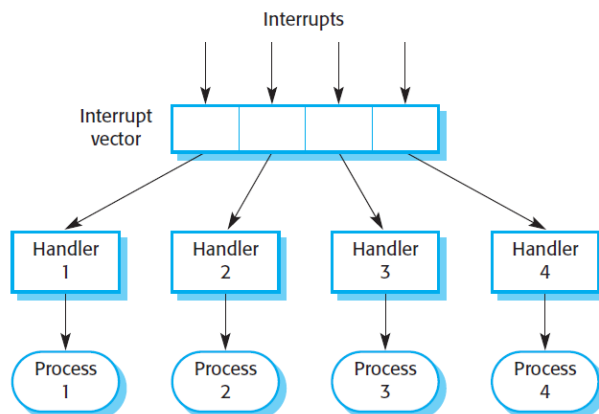
Event-based models are those in which each sub-system can respond to externally generated events from other subsystems or the system's environment. It is a system driven by externally generated events where the timing of the events is out with the control of the subsystems which process the event.

Event-based models are classified into broadcast and interrupt-driven models.


1. Broadcast models: In these models, an event is, in principle, broadcast to all sub-systems. Any sub-system, which is designed to handle that event, responds to it.



2. Interrupt-driven models: These are exclusively used in real-time systems where an interrupt handler detects external interrupts. They are then passed to some other component for processing.



10. What is software design? Explain the various principles and design concepts of software design.

 **hide solution**
asked in 2073

Solution

Software design is the process to transform the user requirements into some suitable form, which helps the programmer in software coding and implementation. During the software design phase, the design document is produced, based on the customer requirements as documented in the SRS document. Hence the aim of this phase is to transform the SRS document into the design document.

Following are the principals and design concept of software design:

1. Software design should correspond to the analysis model: Often a design element corresponds to many requirements, therefore, we must know how the design model satisfies all the requirements represented by the analysis model.

2. Choose the right programming paradigm: A programming paradigm describes the structure of the software system. Depending on the nature and type of application, different programming paradigms such as procedure oriented, object-oriented, and prototyping paradigms can be used. The paradigm should be chosen keeping

constraints in mind such as time, availability of resources and nature of user's requirements.

3. Software design should be uniform and integrated: Software design is considered uniform and integrated, if the interfaces are properly defined among the design components. For this, rules, format, and styles are established before the design team starts designing the software.

4. Software design should be flexible: Software design should be flexible enough to adapt changes easily. To achieve the flexibility, the basic design concepts such as abstraction, refinement, and modularity should be applied effectively.

5. Software design should ensure minimal conceptual (semantic) errors: The design team must ensure that major conceptual errors of design such as ambiguousness and inconsistency are addressed in advance before dealing with the syntactical errors present in the design model.


6. Software design should be structured to degrade gently: Software should be designed to handle unusual changes and circumstances, and if the need arises for termination, it must do so in a proper manner so that functionality of the software is not affected.

7. Software design should represent correspondence between the software and real-world problem: The software design should be structured in such away that it always relates with the real-world problem.

8. Software reuse: Software components should be designed in such a way that they can be effectively reused to increase the productivity.

12. Write short notes on (any two):

- a. DFD
- b. Data dictionary
- c. Estimation techniques

 **Hide solution**
asked in 2073

Solution

a) **DFD**

A **data flow diagram (DFD)** is a tool that depicts the flow of data through a system and the work or processing performed by that system. It shows how data enters and leaves the system, what changes the information, and where data is stored.

The objective of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communication tool between a system analyst and any person who plays a part in the order that acts as a starting point for redesigning a system.

Data Flow Diagrams are either Logical or Physical.

- **Logical DFD** - This type of DFD concentrates on the system process, and flow of data in the system. For example in a Banking software system, how data is moved between different entities.
- **Physical DFD** - This type of DFD shows how the data flow is actually implemented in the system. It is more specific and close to the implementation.

b) **Data Dictionary**

It is a structured repository of data. Although we give descriptive names to the data flows and data stores in a DFD, it does not give the details. Hence to keep the details of the contents of data flows and data stores we require a Data Dictionary. Data Dictionary is a structured repository of data. It clearly documents the list of contents of all data flows and data stores. The three concepts to be defined are:

1. Data Elements: This is the smallest unit of data. Further decomposition is not possible. For example, STUDENT_NAME. Each data element has a **data type** that defines what class of data can be stored in that data element and **domain** that defines what values a data element can legitimately take on.

2. Data Structure: This is a group of Data Elements which together form as a single unit of a data flow.

3. Data flows and Data stores: Data flows are data structures in motion. Data flows can be described in terms of three types of data structures: **sequence** (a group of data elements that occur one after another), **selection** (one or more data elements from a set of data elements), and **repetition** (one or more data elements). Data Stores are data structures in store.

c) **Estimation Techniques**

The Estimation is prediction or a rough idea to determine how much effort would take to complete a defined task. Here the effort could be time or cost. Some estimation techniques are:

- Expert Judgement
- COCOMO model
- Functional point method