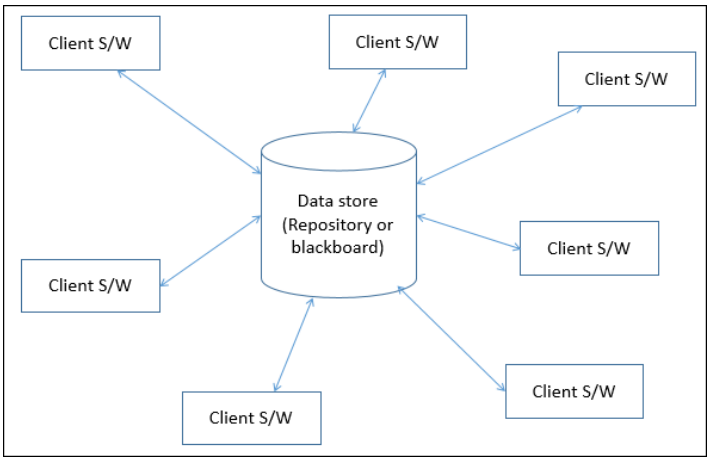
Name: Subhendu Maji Reg. No.: 18ETCS002121 CSE – C section

SDF – Self Assessment 01

Question 1. Differentiate Data Centric Architecture and Data Flow Architecture with an example.

Solution 1.

Data Centric Architecture	**Data Flow Architecture**
In data-centered architecture, the data is centralized and accessed frequently by other components, which modify data. The main purpose of this style is to achieve integrality of data. Data-centered architecture consists of different components that communicate through shared data repositories. The components access a shared data structure and are relatively independent, in that, they interact only through the data store.	In data flow architecture, the whole software system is seen as a series of transformations on consecutive pieces or set of input data, where data and operations are independent of each other. In this approach, the data enters into the system and then flows through the modules one at a time until they are assigned to some final destination (output or a data store).
The most well-known examples of the data-centered architecture is a database architecture, in which the common database schema is created with data definition protocol – for example, a set of related tables with fields and data types in an RDBMS. Another example of data-centered architectures is the web architecture which has a common data schema (i.e. meta-structure of the Web) and follows hypermedia data model and processes communicate through the use of shared web-based data services.  	The connections between the components or modules may be implemented as I/O stream, I/O buffers, piped, or other types of connections. The data can be flown in the graph topology with cycles, in a linear structure without cycles, or in a tree type structure. The main objective of this approach is to achieve the qualities of reuse and modifiability. It is suitable for applications that involve a well-defined series of independent data transformations or computations on orderly defined input and output such as compilers and business data processing applications. There are three types of execution sequences between modules− • Batch sequential • Pipe and filter or non-sequential pipeline mode • Process control

Types of Components

There are two types of components −

- A **central data** structure or data store or data repository, which is responsible for providing permanent data storage. It represents the current state.
- A **data accessor** or a collection of independent components that operate on the central data store, perform computations, and might put back the results.

Interactions or communication between the data accessors is only through the data store. The data is the only means of communication among clients. The flow of control differentiates the architecture into two categories −

- Repository Architecture Style
- Blackboard Architecture Style

Repository Architecture Style

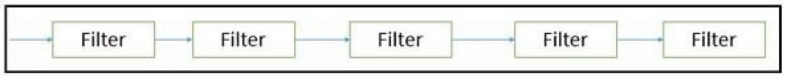
In Repository Architecture Style, the data store is passive and the clients (software components or agents) of the data store are active, which control the logic flow. The participating components check the data-store for changes.

- The client sends a request to the system to perform actions (e.g. insert data).
- The computational processes are independent and triggered by incoming requests.
- If the types of transactions in an input stream of transactions trigger selection of processes to execute, then it is traditional database or repository architecture, or passive repository.
- This approach is widely used in DBMS, library information system, the interface repository in CORBA, compilers and CASE (computer aided software engineering) environments.

Batch Sequential

Batch sequential is a classical data processing model, in which a data transformation subsystem can initiate its process only after its previous subsystem is completely through −

- The flow of data carries a batch of data as a whole from one subsystem to another.
- The communications between the modules are conducted through temporary intermediate files which can be removed by successive subsystems.
- It is applicable for those applications where data is batched, and each subsystem reads related input files and writes output files.
- Typical application of this architecture includes business data processing such as banking and utility billing.



Advantages

- Provides simpler divisions on subsystems.
- Each subsystem can be an independent program working on input data and producing output data.

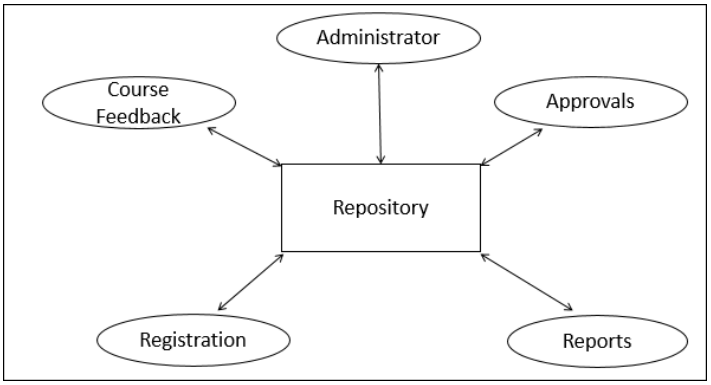
Disadvantages

- Provides high latency and low throughput.
- Does not provide concurrency and interactive interface.
- External control is required for implementation.

Pipe and Filter Architecture

This approach lays emphasis on the incremental transformation of data by successive component. In this approach, the flow of data is driven by data and the whole system is decomposed into components of data source, filters, pipes, and data sinks.

The connections between modules are data stream which is first-in/first-out buffer that can be stream of bytes, characters, or any



Advantages

- Provides data integrity, backup and restore features.
- Provides scalability and reusability of agents as they do not have direct communication with each other.
- Reduces overhead of transient data between software components.

Disadvantages

- It is more vulnerable to failure and data replication or duplication is possible.
- High dependency between data structure of data store and its agents.
- Changes in data structure highly affect the clients.
- Evolution of data is difficult and expensive.
- Cost of moving data on network for distributed data.

Blackboard Architecture Style

In Blackboard Architecture Style, the data store is active and its clients are passive. Therefore, the logical flow is determined by the current data status in data store. It has a blackboard component, acting as a central data repository, and an internal representation is built and acted upon by different computational elements.

other type of such kind. The main feature of this architecture is its concurrent and incremented execution.

Filter

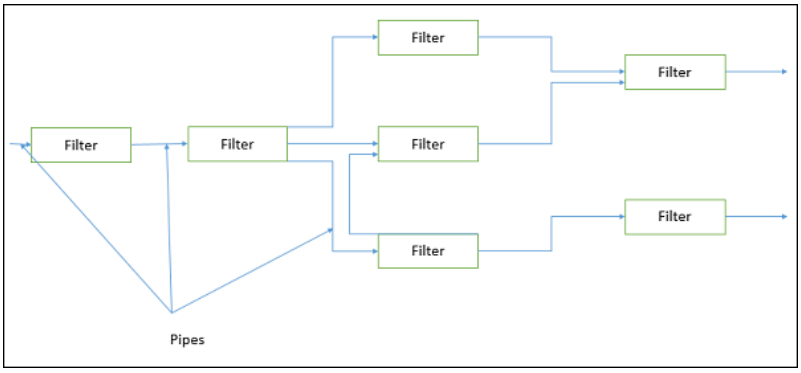
A filter is an independent data stream transformer or stream transducers. It transforms the data of the input data stream, processes it, and writes the transformed data stream over a pipe for the next filter to process. It works in an incremental mode, in which it starts working as soon as data arrives through connected pipe. There are two types of filters – **active filter** and **passive filter**.

Active filter

Active filter lets connected pipes to pull data in and push out the transformed data. It operates with passive pipe, which provides read/write mechanisms for pulling and pushing. This mode is used in UNIX pipe and filter mechanism.

Passive filter

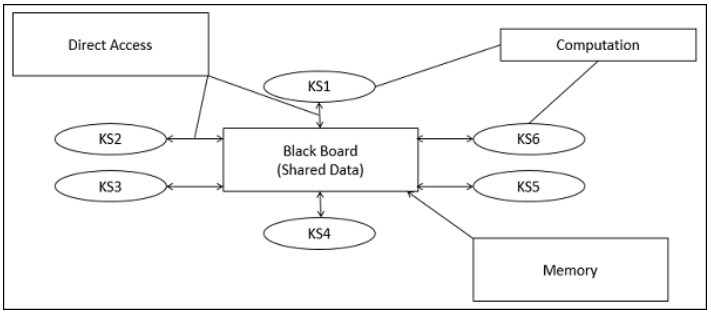
Passive filter lets connected pipes to push data in and pull data out. It operates with active pipe, which pulls data from a filter and pushes data into the next filter. It must provide read/write mechanism.



Advantages

- Provides concurrency and high throughput for excessive data processing.
- Provides reusability and simplifies system maintenance.
- Provides modifiability and low coupling between filters.
- Provides simplicity by offering clear divisions between any two filters connected by pipe.

- A number of components that act independently on the common data structure are stored in the blackboard.
- In this style, the components interact only through the blackboard. The data-store alerts the clients whenever there is a data-store change.
- The current state of the solution is stored in the blackboard and processing is triggered by the state of the blackboard.
- The system sends notifications known as **trigger** and data to the clients when changes occur in the data.
- This approach is found in certain AI applications and complex applications, such as speech recognition, image recognition, security system, and business resource management systems etc.
- If the current state of the central data structure is the main trigger of selecting processes to execute, the repository can be a blackboard and this shared data source is an active agent.
- A major difference with traditional database systems is that the invocation of computational elements in a blackboard architecture is triggered by the current state of the blackboard, and not by external inputs.



Advantages

- Provides scalability which provides easy to add or update knowledge source.
- Provides concurrency that allows all knowledge sources to work in parallel as they are independent of each other.
- Supports experimentation for hypotheses.
- Provides flexibility by supporting both sequential and parallel execution.

Disadvantages

- Not suitable for dynamic interactions.
- A low common denominator is needed for transmission of data in ASCII formats.
- Overhead of data transformation between filters.
- Does not provide a way for filters to cooperatively interact to solve a problem.
- Difficult to configure this architecture dynamically.

Pipe

Pipes are stateless and they carry binary or character stream which exist between two filters. It can move a data stream from one filter to another. Pipes use a little contextual information and retain no state information between instantiations.

Process Control Architecture

It is a type of data flow architecture where data is neither batched sequential nor pipelined stream. The flow of data comes from a set of variables, which controls the execution of process. It decomposes the entire system into subsystems or modules and connects them.

Types of Subsystems

A process control architecture would have a **processing unit** for changing the process control variables and a **controller unit** for calculating the amount of changes.

A controller unit must have the following elements −

- **Controlled Variable** − Controlled Variable provides values for the underlying system and should be measured by sensors. For example, speed in cruise control system.
- **Input Variable** − Measures an input to the process. For example, temperature of return air in temperature control system
- **Manipulated Variable** − Manipulated Variable value is adjusted or changed by the controller.

- Supports reusability of knowledge source agents.

Disadvantages

- The structure change of blackboard may have a significant impact on all of its agents as close dependency exists between blackboard and knowledge source.
- It can be difficult to decide when to terminate the reasoning as only approximate solution is expected.
- Problems in synchronization of multiple agents.
- Major challenges in designing and testing of system.

- **Process Definition** − It includes mechanisms for manipulating some process variables.
- **Sensor** − Obtains values of process variables pertinent to control and can be used as a feedback reference to recalculate manipulated variables.
- **Set Point** − It is the desired value for a controlled variable.
- **Control Algorithm** − It is used for deciding how to manipulate process variables.

Application Areas

Process control architecture is suitable in the following domains −

- Embedded system software design, where the system is manipulated by process control variable data.
- Applications, which aim is to maintain specified properties of the outputs of the process at given reference values.
- Applicable for car-cruise control and building temperature control systems.
- Real-time system software to control automobile anti-lock brakes, nuclear power plants, etc.

Question 2. Explain layered architectural pattern with an example.

Solution 2.

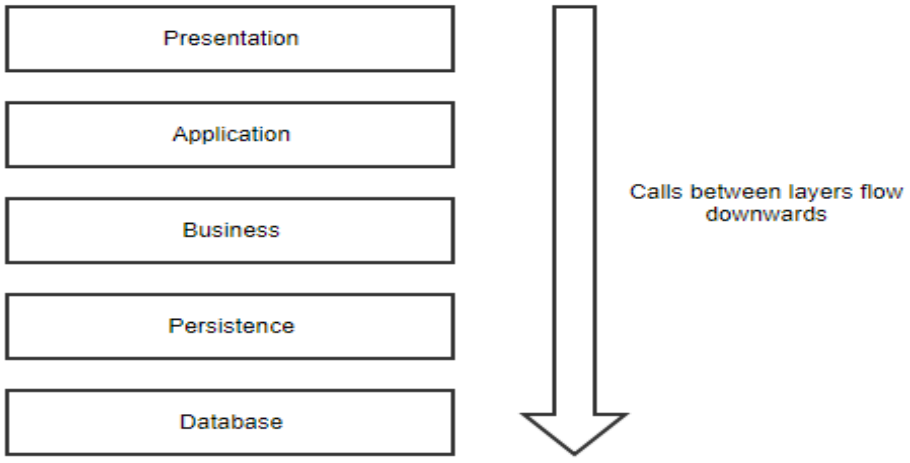
Layered architecture patterns are n-tiered patterns where the components are organized in horizontal layers. This is the traditional method for designing most software and is meant to be self-independent. This means that all the components are interconnected but do not depend on each other.

There isn't a predefined number of layers, but these are the ones you see most often:

- Presentation or UI layer
- Application layer
- Business or domain layer
- Persistence or data access layer
- Database layer

In some cases, the business layer and persistence layer are combined into a single business layer, particularly when the persistence logic (e.g., SQL or HSQL) is embedded within the business layer components. Thus, smaller applications may have only three layers, whereas larger and more complex business applications may contain five or more layers.

The idea is that the user initiates a piece of code in the presentation layer by performing some action (e.g. clicking a button). The presentation layer then calls the underlying layer, i.e. the application layer. Then we go into the business layer and finally, the persistence layer stores everything in the database. So higher layers are dependent upon and make calls to the lower layers.



You will see variations of this, depending on the complexity of the applications. Some applications might omit the application layer, while others add a caching layer. It's even possible to merge two layers into one.

Layer Responsibility

As mentioned, each layer has its own responsibility. The presentation layer contains the graphical design of the application, as well as any code to handle user interaction. You shouldn't add logic that is not specific to the user interface in this layer.

The business layer is where you put the models and logic that is specific to the business problem you are trying to solve.

The application layer sits between the presentation layer and the business layer. On the one hand, it provides an abstraction so that the presentation layer doesn't need to know the business layer. In theory, you could change the technology stack of the presentation layer without changing anything else in your application (e.g. change from WinForms to WPF). On the other hand, the application layer provides a place to put certain coordination logic that doesn't fit in the business or presentation layer.

Finally, the persistence layer contains the code to access the database layer. The database layer is the underlying database technology (e.g. SQL Server, MongoDB). The persistence layer is the set of code to manipulate the database: SQL statements, connection details, etc.

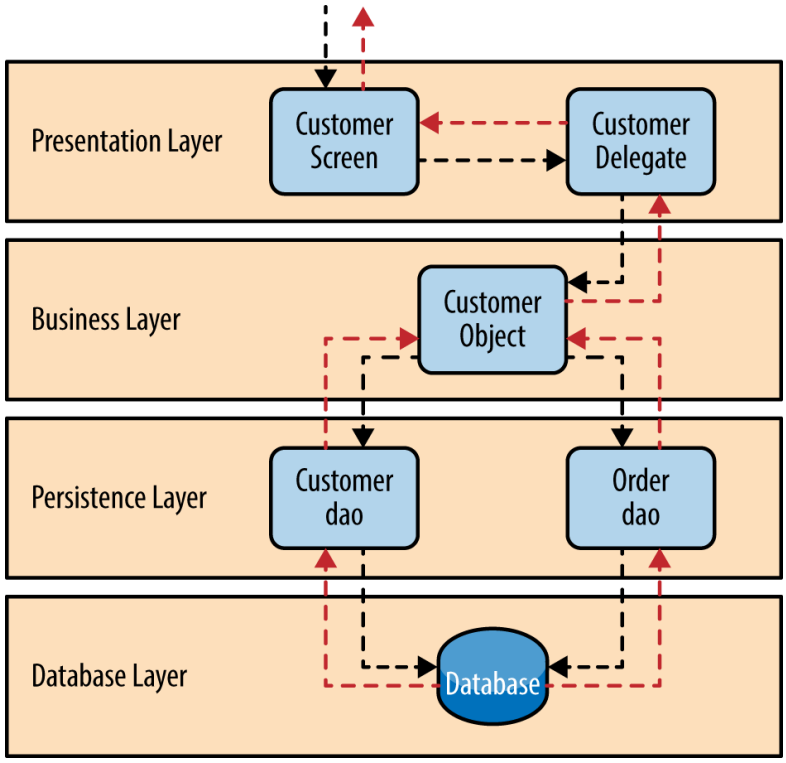
Advantages

- Most developers are familiar with this pattern.
- It provides an easy way of writing a well-organized and testable application.

Disadvantages

- It tends to lead to monolithic applications that are hard to split up afterward.
- Developers often find themselves writing a lot of code to pass through the different layers, without adding any value in these layers. If all you are doing is writing a simple CRUD application, the layered pattern might be overkill for you.

Example Scenario for Layers:



Presentation layer

Presentation of the web pages, UI forms and end user interracting API's.

Business layer

The logic behind the accessibility, security and authentication happens in this layer. This includes ESB (Enterprise Service Buses), middle ware and other various request interceptors to perform validations.

Persistent layer

This is the presentation layer for the Data. This includes the DAO (Data Access Object) presentation, ORM (Object Relational Mappings) and Other modes of presenting persistent data in the application level. In more meaningful words this demonstrates the persistent data in RAM. Which usually stays in Disks at the below layer.

Database layer

Simple Databases expanding up to SANs (Storage Area Networks)

Best for:

- New applications that need to be built quickly
- Enterprise or business applications that need to mirror traditional IT departments and processes
- Teams with inexperienced developers who don't understand other architectures yet
- Applications requiring strict maintainability and testability standards

Question 3. Discuss the working of MVC architecture with a neat diagram.

***Solution 3*. Model View Controller (MVC)**

A software architecture, considered as an architectural pattern used in software engineering. It is a subset of 3-tired architecture. MVC architecture first discussed in 1979 by Trygve Reenskaug.

Features:

- MVC stands for Model-View-Controller.
- MVC architecture separated an application into three main components: model, view and controller.
- It is a software architectural design for implementing user interfaces on computers and is a standard design pattern.
- MVC architecture helps to write better organized and more maintainable code.
- This architecture is used and extensively tested over multiple languages and generations of programmers.
- MVC is popular among various major programming languages such as Java, PHP, ASP.NET etc.
- MVC is being used as the powerful framework for building web applications using MVC pattern.
- MVC separation helps to manage complex applications. It is the main advantage of separation and also simplifies the team development.
- MVC architecture isolates the application logic from the user interface layer and supports the separation of concerns.
- The controller receives all the request for the application and then works with the Model to prepare the data needed by the View.
- The View uses the data prepared by the Controller to generate the final presentable response.
- MVC is a highly testable, extensible and pluggable framework.

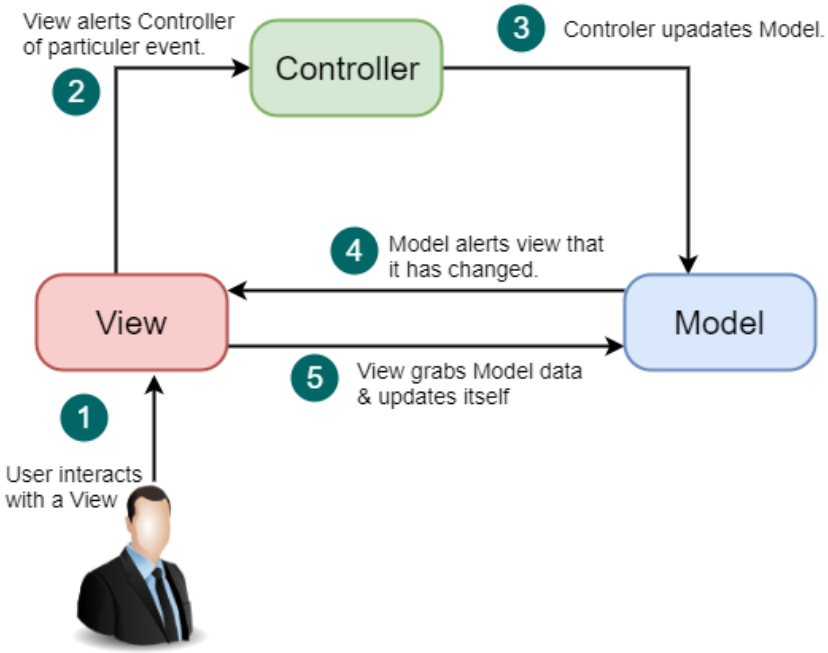


Figure 1 MVC Architecture

Following are the three components of Model-View-Controller:

1. Model
2. View
3. Controller

Components	Description
Model	• Model is the lowest level of the pattern responsible for maintaining the data. • It is a central component of MVC architecture. • It manages the data, logic and constraints of an application. • It captures the behavior of an application domain problem. • It is the domain-specific software implementation of the application's central structure. • If there is any change in its state then it gives notification to its associated view to produce updated output and the controller to change the available set of commands. • It is an independent user interface.
View	• View is responsible for displaying all or a portion of the data to the user. • It represents any output of information in a graphical form such as diagram or chart. • View consists of presentation components which provide the visual representations of data. • View formats and presents the data from model to user.
Controller	• Controller controls the interactions between the Model and View. • It accepts an input and converts it into the commands for model or view. • Controller acts as an interface between the associated models, views and the input devices. • It sends the commands to the model to update the model's state and to its associated view to change the view's presentation of the model.

Advantages of MVC Architecture

- Model-View-Controller Architecture supports rapid and parallel development.
- This model has ability to provide multiple views.
- It supports asynchronous technique which helps developer to develop an application that loads very fast.
- It manages complexity of large applications and keep it well organized because of MVC's separated nature.
- In MVC architecture, modification does not affect the entire model.
- It returns the data without any formatting.
- It is easy to maintain and gives good productivity.
- It is easy to plug-in new or replace interface views.

Disadvantages of MVC Architecture

- MVC architecture is not suitable for agent-oriented applications such as mobile interactive and robotics applications.
- It makes any data model expensive because multiple pairs of controllers and views based on the same data model.