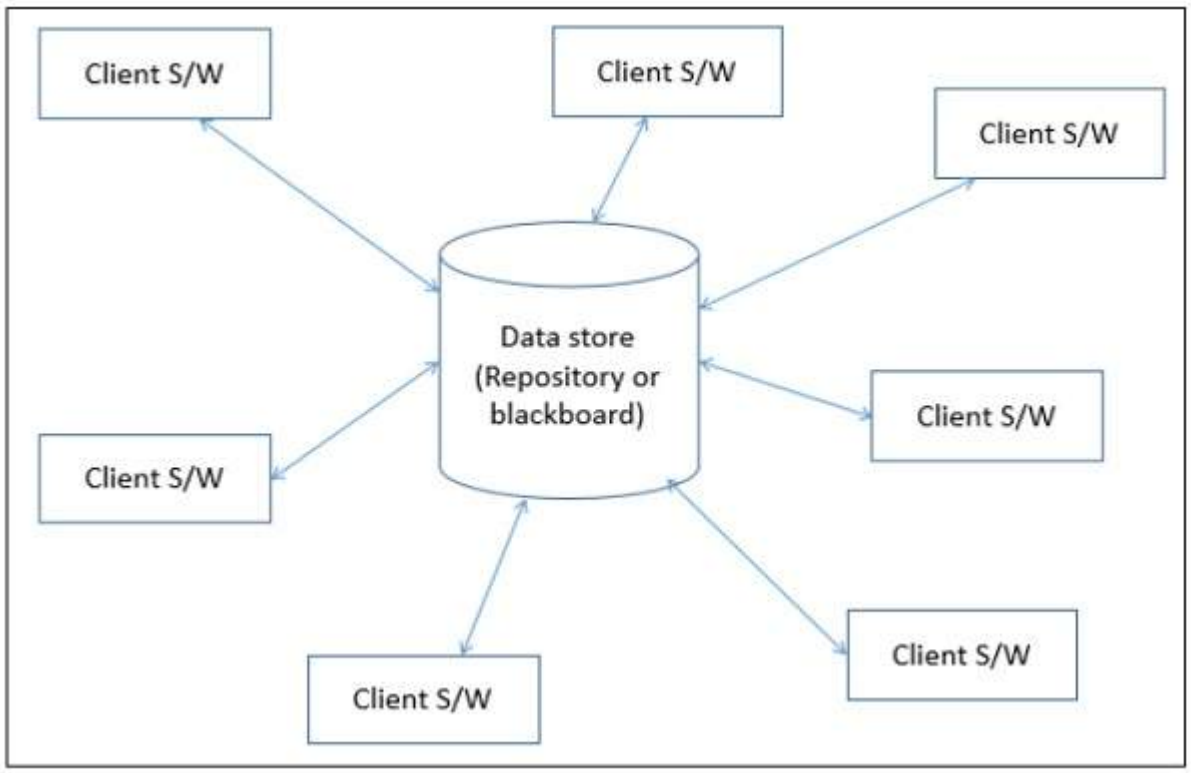
DATA-CENTERED ARCHITECTURE

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.

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 − structureoftheWeb* and follows hypermedia data model and processes communicate through the use of shared web-based data services.



Types of Components

There are two types of components −

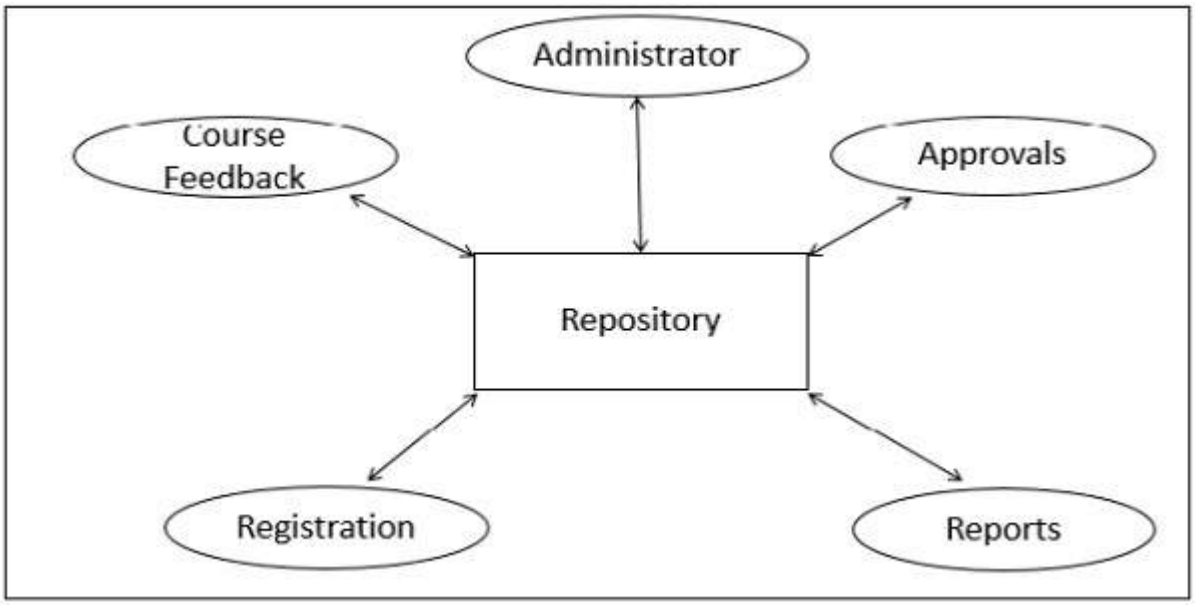
- A **central data** structure or data store or data repository, which is responsible for providing permanent data storage. It represents the current state.
- A **data accessor** or a collection of independent components that operate on the central data store, perform computations, and might put back 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 as **Repository Architecture Style** and **Blackboard Architecture Style**. A brief detail about both the categories is given below −

Repository Architecture Style

In Repository Architecture Style, the data store is passive and the clients *softwarecomponentsoragents* of the data store are active, which control the logic flow. The participating components check the data-store for changes.

A client sends a request to the system to perform actions *e. g. insertdata*. 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 *computeraidedsoftwareengineering* environments.



Advantages

Repository Architecture Style has following 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

Because of being more vulnerable to failure and data replication or duplication, Repository Architecture Style has following disadvantages −

- 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.

Further, 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 changes. The current state of the solution is stored in the blackboard and processing is triggered by the state of the blackboard.

When changes occur in the data, the system sends the notifications known as **trigger** and data to the clients. This approach is found in certain AI applications and complex applications, such as

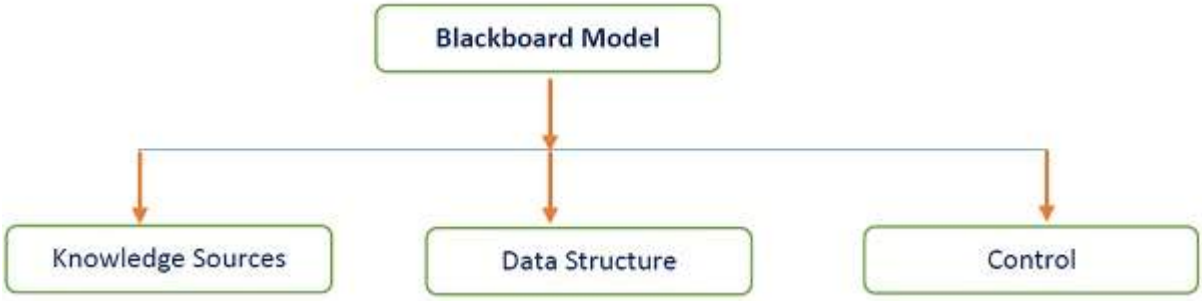
speech recognition, image recognition, security system, and business resource management systems etc.

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.

Parts of Blackboard Model

The blackboard model is usually presented with three major parts −

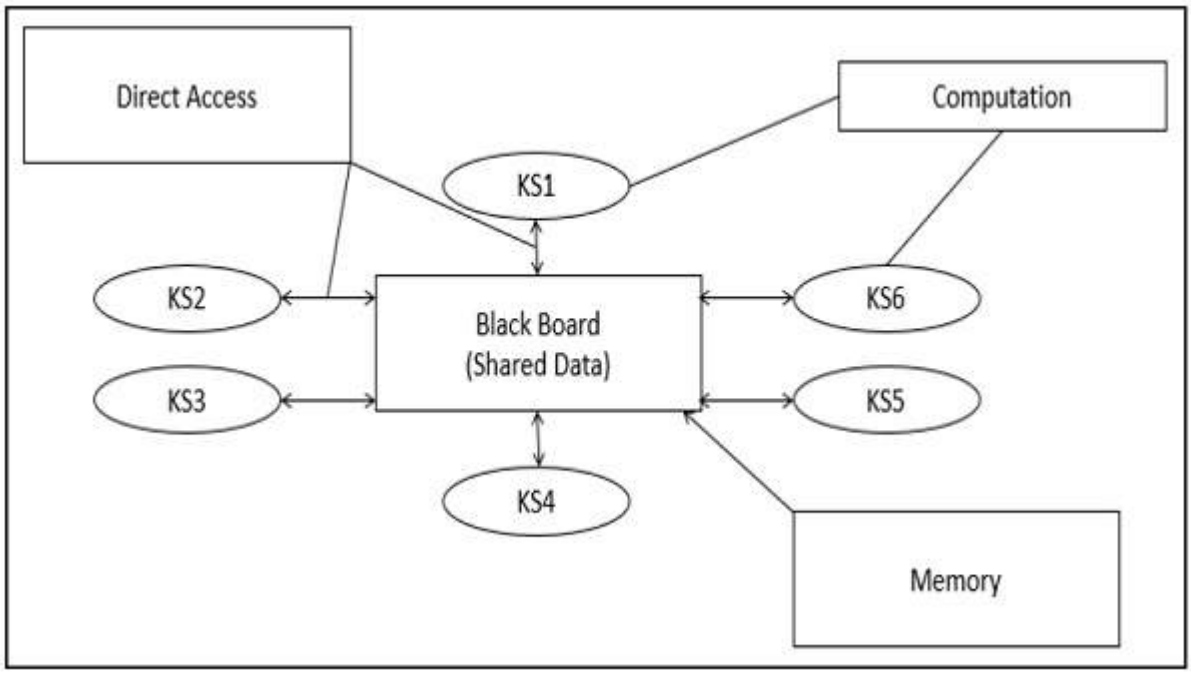


Knowledge Sources *KS*

Knowledge Sources, also known as **Listeners** or **Subscribers** are distinct and independent units. They solve parts of a problem and aggregate partial results. Interaction among knowledge sources takes place uniquely through the blackboard.

Blackboard Data Structure

The problem-solving state data is organized into an application-dependent hierarchy. Knowledge sources make changes to the blackboard that lead incrementally to a solution to the problem.



Control

Control manages tasks and checks the work state.

Advantages

Blackboard Model provides concurrency that allows all knowledge sources to work in parallel as they independent of each other. Its scalability feature facilitates easy steps to add or update knowledge source. Further, it supports experimentation for hypotheses and reusability of knowledge source agents.

Disadvantages

The structural change of blackboard may have a significant impact on all of its agents, as close dependency exists between blackboard and knowledge source. Blackboard model is expected to produce approximate solution; however, sometimes, it becomes difficult to decide when to terminate the reasoning.

Further, this model suffers some problems in synchronization of multiple agents, therefore, it faces challenge in designing and testing of the system.