CSE 470
Software Engineering
Software Architecture

Imran Zahid
Lecturer
Computer Science and Engineering, BRAC University

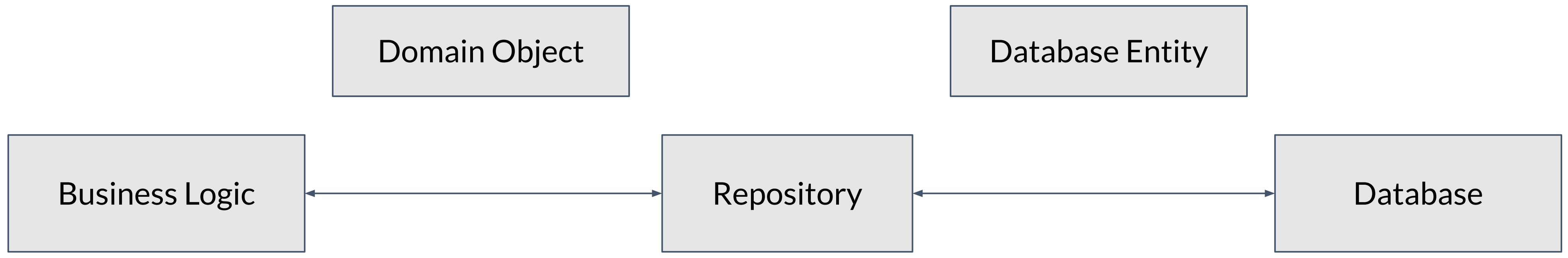
Repository Architecture

Repository Architecture

- We might have multiple subsystems in our software architecture.
- Subsystems must exchange data. This may be done in two ways:
    - Shared data is held in a central database or repository and may be accessed by all sub-systems;
    - Each sub-system maintains its own database and passes data explicitly to other sub-systems.
- When large amounts of data are to be shared, the repository model of sharing is most commonly used as this is an efficient data sharing mechanism.

Repository Architecture

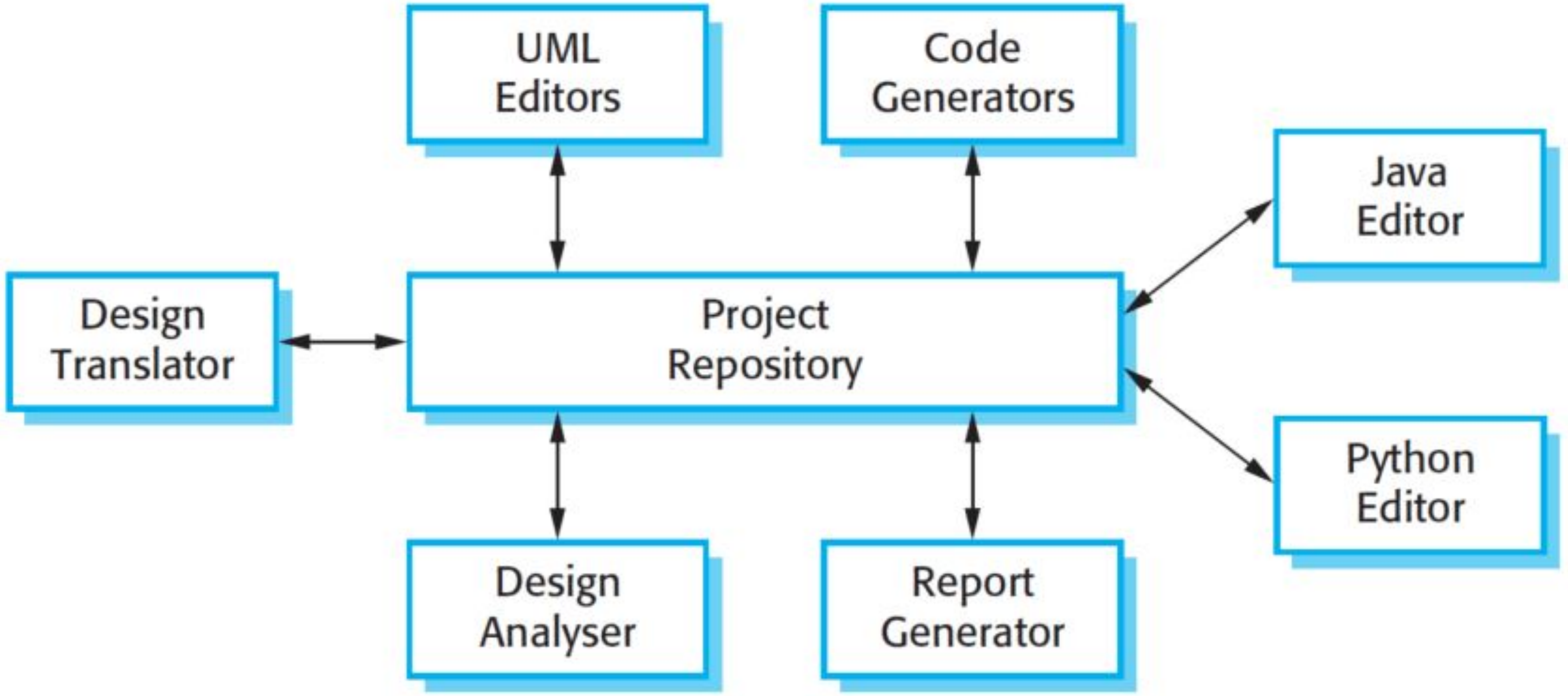
- The Repository Architecture is a design pattern that provides a way to manage data access logic in a centralized location.
- It separates the logic that retrieves the data and maps it to the domain model from the business logic that operates on the model.



Repository Architecture

- Repository mediates between domain and data mapping layers, acting like an in-memory domain object collection.
- Client objects construct query specifications declaratively and submit them to Repository for processing.
- Objects can be added to and removed from the Repository like a simple collection. Repository encapsulates mapping code, executing appropriate operations behind the scenes.

Repository Architecture - Example



A repository architecture for an IDE

Repository Architecture

- All data in a system is managed in a central repository that is accessible to all system components.
- Components do not interact directly, only through the repository.

Repository Architecture - When to Use?

- You should use this pattern when you have a system in which large volumes of information are generated that has to be stored for a long time.
- You may also use it in data-driven systems where the inclusion of data in the repository triggers an action or tool.

Repository Architecture - Advantages

- Components can be independent—they do not need to know of the existence of other components.
- Changes made by one component can be propagated to all components.
- All data can be managed consistently (e.g., backups done at the same time) as it is all in one place.

Repository Architecture - Disadvantages

- The repository is a single point of failure so problems in the repository affect the whole system.
- May be inefficiencies in organizing all communication through the repository.
- Distributing the repository across several computers may be difficult.

Client-Server

Client-Server Architecture

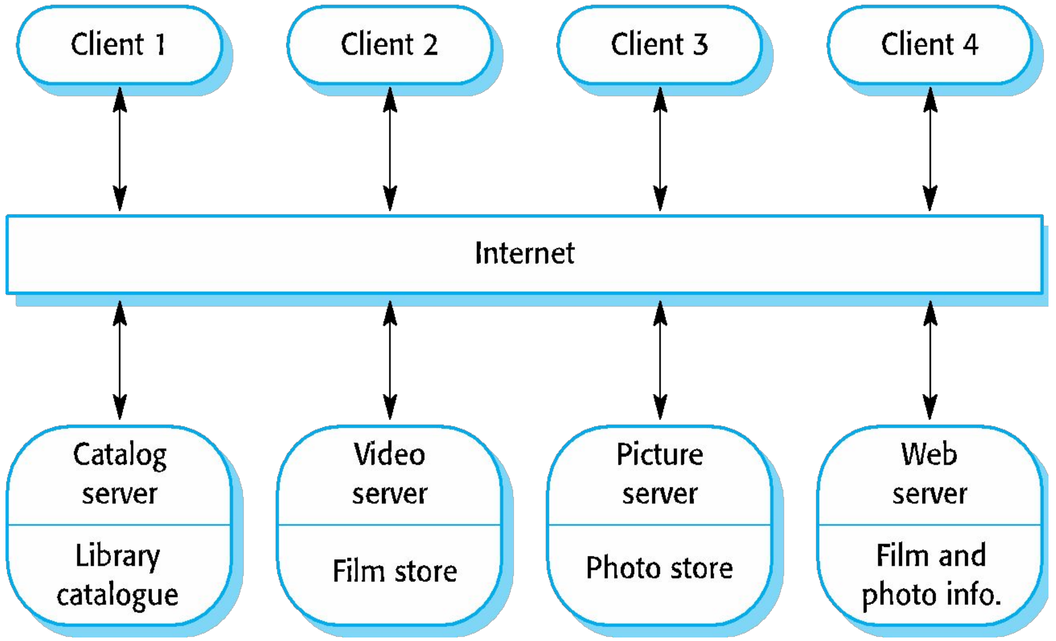
- This pattern consists of two parties; servers and multiple clients.
- The server provides services to multiple client components.
- The client interacts with the user and sends requests to the server, which processes them and provides the necessary information or results.
- Furthermore, the server continues to listen to client requests.

Client-Server Architecture

- The client-server architecture has three components:
    - Set of stand-alone servers which provide specific services such as printing, data management, etc.
    - Set of clients which call on these services.
    - Network which allows clients to access servers.
- In a client–server architecture, the functionality of the system is organized into services, with each service delivered from a separate server. Clients are users of these services and access servers to make use of them.

Example

- Online applications such as email, document sharing and banking.
- Here is an example of a film and video/DVD library organized as a client–server system.



Client-Server Architecture - When to Use?

- Used when data in a shared database has to be accessed from a range of locations.
- Because servers can be replicated, may also be used when the load on a system is variable.

Client-Server Architecture - Advantages

- The principal advantage of this model is that servers can be distributed across a network.
- General functionality (e.g., a printing service) can be available to all clients and does not need to be implemented by all services.

Client-Server Architecture - Disadvantages

- Each service is a single point of failure so susceptible to denial of service attacks or server failure.
- Performance may be unpredictable because it depends on the network as well as the system.
- There may be management problems if servers are owned by different organizations.

BRAC UNIVERSITY
Inspiring Excellence

Thank you