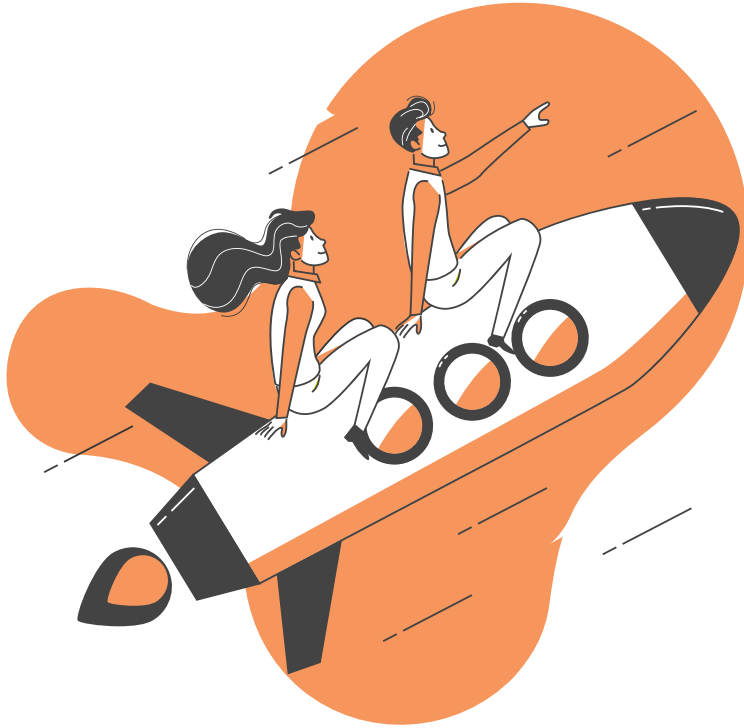# SOFTWARE ARCHITECTURE

CSE470 slide presented by
A.M.ESFAR-E-ALAM
AFRINA KHATUN
DR.ZAVID PARVEZ
HOSSAIN ARIF

01
REPOSITORY PATTERN

02
CLIENT SERVER PATTERN

03
PIPE AND FILTER PATTERN

04
Advantages and Disadvantages

BRAC
UNIVERSITY
Inspiring Excellence

REPOSITORY PATTERN

REPOSITORY

Before going into repository design pattern let's first understand what is repository:

- In generic terms repository simply means a place where things are or can be stored.
- This term has wide array of use in computer science for example code repository.
- We are all familiar with GITHUB. it's a code repository

DEFINING REPOSITORY PATTERN

So now we have a idea of what repository means, let's think how that is connected to software architecture:

- Here we will have multiple sub systems.
- Sub-systems 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.
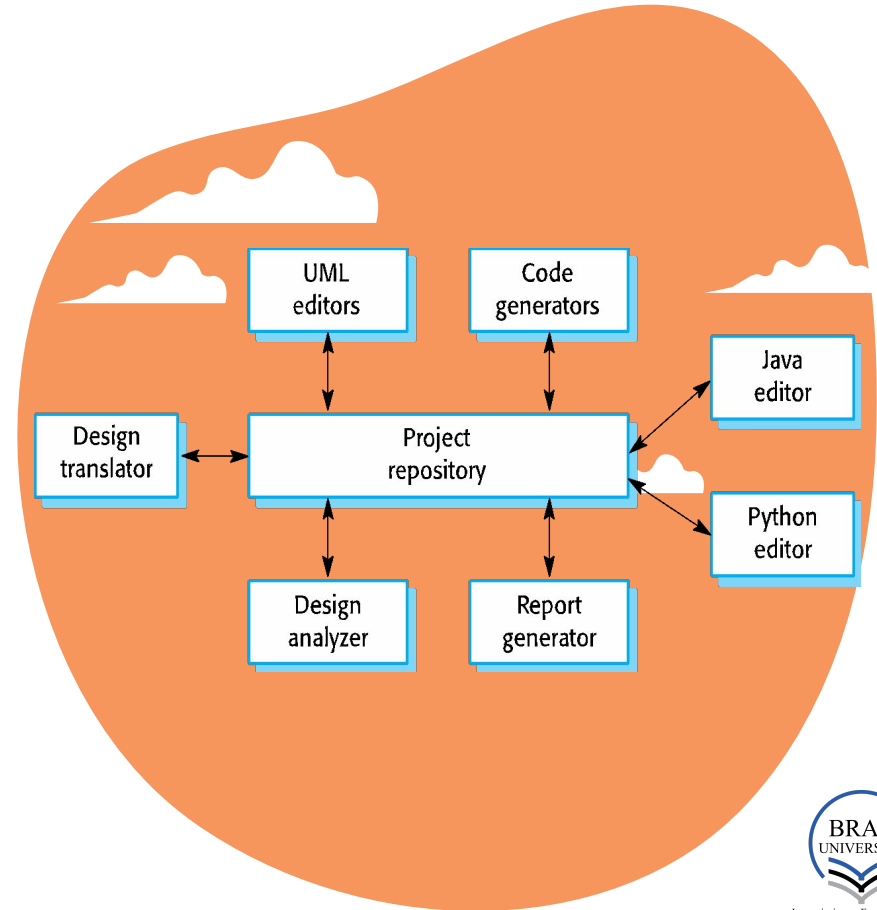- So lets define repository pattern formally

- "A Repository mediates between the 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 satisfaction. Objects can be added to and removed from the Repository, as they can from a simple collection of objects, and the mapping code encapsulated by the Repository will carry out the appropriate operations behind the scenes."

—FORMAL DEFINITION

EXAMPLE!

- Let's think you are developing an IDE like 'eclipse'.What will you have there:

    - You will have editor for different language like java,c++.python
    - You will have code generator.Auto complete system
    - UML editor
    - Translator to compile the code etc.

- So here in the IDE where the components use a repository of system design information.

- Each software tool generates information which is then available for use by other tools.

- And, sub-components do not communicate with each other, means they are independent.

- Have a look at the figure for the example

Summarising

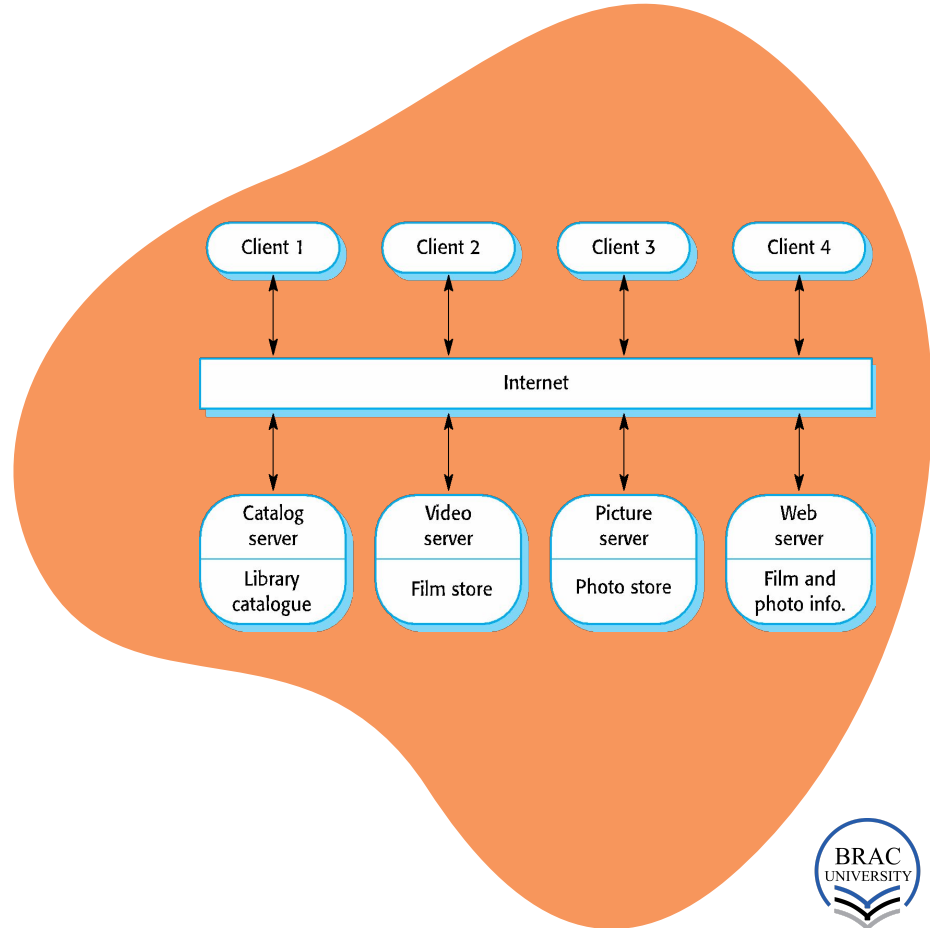| Name	Repository
Description	All data in a system is managed in a central repository that is accessible to all system components. Components do not interact directly, only through the repository.
When used	You should use this pattern when you have a system in which large volumes of information are generated 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.
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.
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 – latency and processing overhead. • Distributing the repository across several computers may be difficult. (Scalability issue, can be solved by synchronization, but, complex and costly. Also, lacks fault tolerance if distributed)

2. Client-server pattern

Client-server pattern

- This pattern consists of two parties; **a server and multiple clients.**
- The server component will provide services to multiple client components.
- Clients request services from the server and the server provides relevant services to those clients.
- Furthermore, the server continues to listen to client requests.
- 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.

Client Server Architecture

Internet
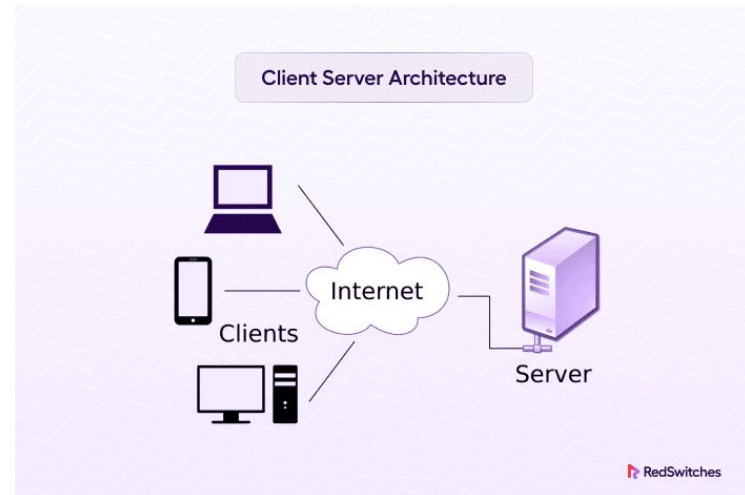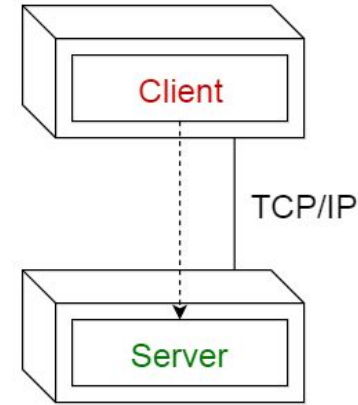
Clients

Server

EXAMPLE

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

Summarizing Client-Server Architecture

| Name	Client-server
Description	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.
When used	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 and can be resolved by a load balancer.
Advantages	• The principal advantage of this model is that servers can be distributed across a network. • Cost efficient as requires less maintenance cost and Data recovery is possible. • Can be deployed to the cloud, hosted and managed by third-parties (AWS, Azure) which offloads physical infrastructure burden and easy to manage.
Disadvantages	• Each service is a single point of failure so susceptible to denial of service (DoS) attacks or server failure. • Performance may be unpredictable because it depends on the network as well as the system. • May be management problems if servers are owned by different organizations.

BRAC
UNIVERSITY

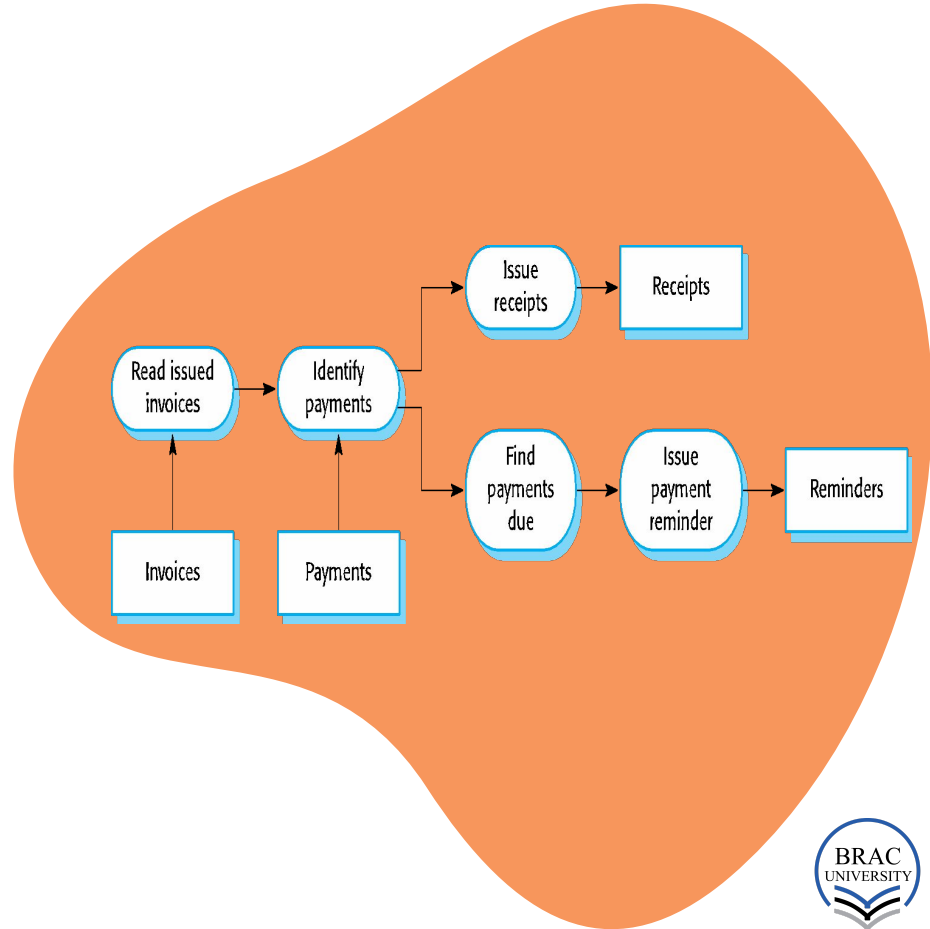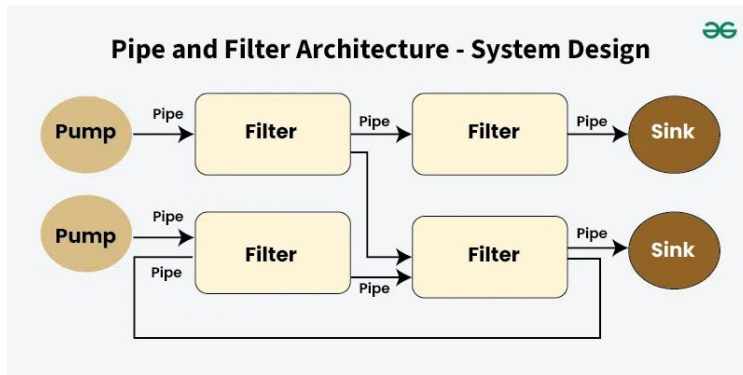Inspiring Excellence

3. Pipe-filter pattern

What is it about?

This pattern can be used to structure systems which produce and process a stream of data. Each processing step is enclosed within a filter component. Data to be processed is passed through pipes. These pipes can be used for buffering or for synchronization purposes.
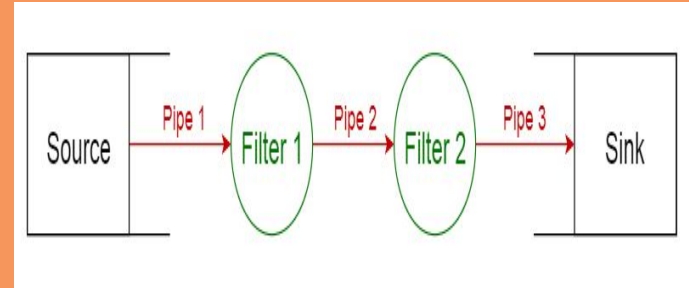
Usage

- **Compilers. The consecutive filters perform lexical analysis, parsing, semantic analysis, and code generation.**
- **Figure beside is an example of the pipe and filter architecture used in a payments system**

Pipe and Filter Architecture - System Design

SUMMARIZING PIPE-FILTER

Name	**Pipe and filter**
Description	The processing of the data in a system is organized so that each processing component (filter) is discrete and carries out one type of data transformation. The data flows (as in a pipe) from one component to another for processing.
When used	Commonly used in data processing applications (both batch- and transaction-based) where inputs are processed in separate stages to generate related outputs.
Advantages	• Easy to understand and supports transformation reuse. • Workflow style matches the structure of many business processes. • Evolution by adding transformations is straightforward. • Can be implemented as either a sequential or concurrent system.
Disadvantages	• The format for data transfer has to be agreed upon between communicating transformations. • Each transformation must parse its input and unparse its output to the agreed form. • This increases system overhead and may mean that it is impossible to reuse functional transformations that use incompatible data structures.

BRAC
UNIVERSITY

Inspiring Excellence

Thank you

Read supplementary slides
More Reading:

https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013
https://medium.com/@tkchandravadhana/pipe-and-filter-architecture-style-f0e317df4c85

BRAC
UNIVERSITY

Inspiring Excellence