Architectural Pattern

In software engineering, an architectural pattern is a general, reusable solution to a commonly occurring architectural problem in software design. Architectural patterns provide a way to abstract and generalize architectural solutions, making them easier to understand, implement, and maintain. Here are some of the most common types of architectural patterns:

1. Layered architecture: Layered architecture is an architectural pattern that divides an application into a set of horizontal layers, each of which has a specific responsibility and interacts with adjacent layers in a predefined way. Examples of layers include presentation, business logic, and data access layers.

2. Client-server architecture: Client-server architecture is an architectural pattern that divides an application into two parts, a client and a server, which communicate with each other over a network. The server provides services or data to the client, while the client sends requests and displays the results.

3. Model-view-controller (MVC) architecture: MVC architecture is an architectural pattern that divides an application into three parts, a model, a view, and a controller. The model represents the data and business logic of the application, the view represents the user interface, and the controller manages the interaction between the model and the view.

4. Microservices architecture: Microservices architecture is an architectural pattern that divides an application into a set of small, independent services, each of which has a specific responsibility and communicates with other services through well-defined APIs. This pattern is often used for large, complex, and distributed applications.

5. Event-driven architecture: Event-driven architecture is an architectural pattern that focuses on the flow of events within an application, with each event triggering a specific action or response. This pattern is often used for applications that require real-time processing and analysis of large volumes of data.

6. Service-oriented architecture (SOA): SOA is an architectural pattern that focuses on the use of services, which are self-contained, modular components that provide specific functionality and can be accessed and used by other components or applications.

7. Pipes and Filters architecture: Pipes and Filters is an architectural pattern that divides an application into components called filters, which process data in a pipeline of connected filters. Each filter performs a specific transformation on the input data and passes it on to the next filter in the pipeline. This pattern is often used for applications that require data processing or transformation, such as image or audio processing.

8. Peer-to-Peer (P2P) architecture: P2P is an architectural pattern in which all nodes in a network are both clients and servers, and communicate with each other directly without the need for a central server. This pattern is often used for distributed applications that require high scalability, fault tolerance, and decentralization.

9. Blackboard architecture: Blackboard is an architectural pattern that divides an application into three parts, a blackboard, knowledge sources, and a control component. The blackboard is a central data repository that stores information about the problem being solved, while the knowledge sources are independent components that provide domain-specific knowledge and algorithms. The control component manages the interactions between the blackboard and the knowledge sources. This pattern is often used for applications that require knowledge-based or expert system functionality.

10. Domain-Driven Design (DDD) architecture: DDD is an architectural pattern that focuses on the design and organization of the application's business logic around the core domain of the application. This pattern emphasizes the use of domain models, ubiquitous language, and bounded contexts to ensure that the application's design reflects the real-world problem being solved. This pattern is often used for complex applications with a large and evolving domain.

11. Event sourcing architecture: Event sourcing is an architectural pattern that captures all changes to an application's state as a sequence of events. Each event represents a specific change to the application's state, and is stored in an event log. This pattern allows the application to recreate its state at any point in time by replaying the events in the event log. Event sourcing is often used for applications that require auditability, scalability, and fault tolerance.

12. Hexagonal architecture: Hexagonal architecture, also known as Ports and Adapters architecture or Clean architecture, is an architectural pattern that emphasizes the separation of concerns between the application's core business logic and its external dependencies, such as databases, web services, and user interfaces. This pattern uses a set of interfaces, or ports, to define the interactions between the core logic and the external adapters, allowing the application to be easily adapted to changes in external dependencies. Hexagonal architecture is often used for applications that require flexibility, maintainability, and testability.

13. Cloud-native architecture: Cloud-native architecture is an architectural pattern that is optimized for deployment in cloud environments, such as public or private clouds. This pattern emphasizes the use of containerization, microservices, and automation to achieve high scalability, fault tolerance, and agility. Cloud-native architecture is often used for applications that require rapid development and deployment, and can benefit from the elasticity and scalability of cloud infrastructure.

14. Reactive architecture: Reactive architecture is an architectural pattern that emphasizes the use of asynchronous and non-blocking communication between components, in order to achieve high responsiveness, scalability, and fault tolerance. This pattern uses a set of principles, such as message-driven, elastic, and resilient, to guide the design of the application's components and interactions. Reactive architecture is often used for applications that require high concurrency, real-time processing, and responsiveness.

15. CQRS architecture: CQRS (Command Query Responsibility Segregation) architecture is an architectural pattern that separates the responsibilities of reading and writing data in an application. This pattern uses separate models and interfaces for commands (write operations) and queries (read operations), allowing the application to optimize each aspect independently. CQRS architecture is often used for applications that require high performance, scalability, and flexibility in handling data.