Angular utilizes several design patterns to facilitate the development of robust and maintainable applications. Some of the key design patterns commonly employed in Angular applications include:

1. Component-Based Architecture: This is more of an architectural pattern rather than a design pattern per se. Angular applications are built using components, which are self-contained units of functionality that encapsulate the view and behavior of a portion of the user interface. This promotes reusability, modularity, and maintainability.
2. Dependency Injection (DI): Angular employs the Dependency Injection pattern to manage the dependencies between different components and services. DI enables the creation and injection of dependencies into a component or service, which promotes loose coupling and makes components easier to test and maintain.
3. Observer (subscription)/Observable Pattern: Angular leverages RxJS, a reactive programming library, which is based on the Observer/Observable pattern. Observables represent a stream of data that can be observed over time, and observers subscribe to these observables to react to changes. This pattern is extensively used for handling asynchronous operations, such as HTTP requests and event handling.
4. Facade Pattern: Angular uses the Facade pattern to provide a simple interface for complex subsystems or libraries. For example, Angular's HttpClient service serves as a facade for making HTTP requests, abstracting away the complexity of handling XHR requests directly.
5. Decorator Pattern: Angular decorators, such as @Component, @Injectable, @Directive, @Pipe, etc., are based on the Decorator pattern. Decorators provide a convenient way to add metadata and behavior to classes, methods, and properties at design time.
6. Singleton Pattern: While not explicitly enforced by Angular, the Singleton pattern is often used implicitly for services. Angular's DI system creates a single instance of a service and shares it throughout the application, ensuring that all components and modules access the same instance.