11 Lecture - CS504

Important Subjective

What is the difference between cohesion and coupling in software design? Answer: Cohesion refers to the degree of relatedness within a module, indicating how closely the responsibilities of the module are aligned. Coupling, on the other hand, measures the degree of interdependence between modules. High cohesion and low coupling are desirable design qualities. **Explain the SOLID principles of software design. Answer**: SOLID is an acronym representing five design principles: Single Responsibility Principle (SRP), Open-Closed Principle (OCP), Liskov Substitution Principle (LSP), Interface Segregation Principle (ISP), and Dependency Inversion Principle (DIP). These principles guide software design to achieve modularity, maintainability, and extensibility. **What is the purpose of design patterns in software development? Answer**: Design patterns provide proven solutions to recurring design problems in software development. They promote reusable and modular designs, improve code maintainability, and facilitate communication among developers by providing a common vocabulary for discussing design concepts. **What is the difference between the factory method and abstract factory design patterns? Answer**: The factory method pattern defines an interface for creating objects, but allows subclasses to decide which class to instantiate. In contrast, the abstract factory pattern provides an interface for creating families of related or dependent objects without specifying their concrete classes. **What are the advantages of using the Model-View-Controller (MVC) architectural pattern? Answer**: MVC promotes separation of concerns by dividing an application into three components: the model (data and business logic), the view (user interface), and the controller (handles user input and coordinates model-view interaction). Benefits include modularity, code reusability, and ease of maintenance. **Explain the concept of loose coupling in software design. Answer**: Loose coupling refers to reducing dependencies between software components. It allows changes in one component to have minimal impact on others, making the system more flexible and maintainable. Loose coupling is achieved through well-defined interfaces and dependency injection. **What is the purpose of the decorator design pattern? Answer**: The decorator pattern allows behavior to be added dynamically to an object at runtime by wrapping it in a decorator class. This pattern provides a flexible alternative to subclassing for extending the functionality of individual objects without modifying their original classes. **Describe the concept of inversion of control (IoC) in software design. Answer**: Inversion of Control refers to the shift of control from application code to a framework or container. Instead of instantiating and managing dependencies directly, IoC containers handle the creation and injection of dependencies, promoting loose coupling and modular design. **What is the role of the architect in software design? Answer**: The architect is responsible for designing the overall structure and behavior of a software system. They make high-level design decisions, define architectural patterns, choose appropriate technologies, and ensure that the system meets functional and non-functional requirements. **What are the key considerations when designing for scalability in software systems? Answer**: When designing for scalability, considerations include distributed architecture, horizontal scaling, efficient resource utilization, load balancing, caching strategies, and the ability to handle increasing volumes of data and users while maintaining performance and reliability.