CS101 updated VU Final term Past Paper Last 5 Year
Created by: VU APEX Campus Teachers Team

Questions:

1. **What is cohesion in software engineering, and why is it important?**
Answer: Cohesion in software engineering refers to the internal binding or relatedness of a module's internal parts. It measures the degree to which the elements within a module are functionally related. High intramodule cohesion is essential for simplifying processes during module changes. Logical cohesion is a weaker form, induced by similar logical activities, while functional cohesion is stronger, focusing on a single activity. Achieving functional cohesion often involves isolating subtasks in other modules. In object-oriented designs, entire objects are usually logically cohesive, but designers aim for functional cohesion at the method level within objects.
2. **Explain the concept of information hiding in software engineering.**
Answer: Information hiding involves restricting information about a software system to a specific portion, preventing unnecessary dependencies or effects on other modules. It includes data, data structures, encoding systems, module composition, and procedural unit structures. Information hiding serves as both a design and implementation goal. Design-wise, modules should be designed to minimize external access to their internal information. In implementation, techniques like encapsulation and well-defined control structures reinforce module boundaries. Information hiding is crucial for module validity, protecting against errors during development and misguided maintenance efforts. It aligns with the theme of abstraction, treating modules as "black boxes," emphasizing user focus on larger applications.
3. **How does the modular approach in software development address the lack of prefabricated building blocks?**
Answer: The modular approach in software development addresses the lack of prefabricated building blocks by leveraging the object-oriented programming paradigm. Objects, as self-contained units with well-defined interfaces, serve as reusable building blocks. Objects/classes designed for specific roles can be used in various programs. Inheritance allows customization when needed.

While objects provide small building blocks, the concept of components extends this idea. Components, often based on the object-oriented paradigm, are reusable units of software. Component architectures, like in smartphones, use predefined components displayed graphically, allowing assembly without extensive internal programming. This approach improves integration between applications and efficiently utilizes system resources.

4. **What is the significance of design patterns in software engineering, and how do they contribute to the development process?**

Answer: Design patterns are pre-developed models for solving recurring problems in software design. They play a crucial role in software engineering by offering standardized solutions to common issues. The Adapter pattern, for instance, helps integrate prefabricated modules with incompatible interfaces. Another pattern, the Decorator pattern, manages systems with different combinations of activities. Design patterns help minimize coupling and maximize cohesion, following good design principles. They contribute to software development by providing high-quality, ready-made solutions. Many design patterns originate from diverse fields, with Christopher Alexander's architectural research influencing their development. Today, design patterns are integral to modern software development tools, fostering efficient and reliable solutions.