Software Design and Development

Comprehensive Revision Lecture: Key Concepts and Practical Applications

Introduction to Software Design

- Software design ensures maintainability and scalability.
- Key goals: simplicity, clarity, and adaptability (McGrath et al., 2013).

Importance of Design Principles

- Guides development to reduce technical debt (Beck, 2003).
- Ensures robustness and quality in software.

Single Responsibility Principle (SRP)

- Each class or module has one responsibility.
- Analogy: A chef doesn’t also clean the dining area.

Open-Closed Principle (OCP)

- Modules should be extendable without modification.
- Analogy: Adding a new feature to a phone without altering its existing hardware.

Liskov Substitution Principle (LSP)

- Derived classes should work as substitutes for base classes.
- Analogy: Electric cars can replace fuel cars in a garage.

Interface Segregation Principle (ISP)

- Avoid forcing classes to implement unused interfaces.
- Analogy: A universal remote doesn’t need all buttons for every device.

Dependency Inversion Principle (DIP)

- High-level modules depend on abstractions, not details.
- Analogy: Using an adapter to plug devices into different outlets.

Don’t Repeat Yourself (DRY)

- Avoid redundancy to improve maintainability.
- Stat: 60% of bugs stem from repetitive code (Larman, 2004).

Applying DRY in Projects

- Centralize logic in functions or modules.
- Use templates for repeated structures.

Modularization

- Break systems into independent, cohesive modules.
- Example: Separate billing, inventory, and user management.

Encapsulation

- Restrict access to internal module details.
- Example: Using private methods in classes.

Abstraction

- Simplify complex details behind simple interfaces.
- Example: A `print()` function abstracts printer drivers.

Version Control

- Git tracks changes and enables collaboration.
- Stat: 99% of developers use version control (Chacon & Straub, 2014).

Unit Testing

- Test individual components in isolation.
- Stat: Teams with automated tests deploy 40% faster (Beck, 2003).

Continuous Integration (CI)

- Automate testing with each code change.
- Stat: CI reduces deployment errors by 30%.

Secure Software Design

- Incorporate security at every development stage.
- Stat: 70% of breaches result from poor design (OWASP, 2023).

Emerging Trends

- AI in bug detection and prevention.
- Stat: AI reduces time-to-detection by 50%.

Recap of Key Principles

- SOLID: Core design principles for maintainable code.
- DRY: Avoid redundancy for clarity and efficiency.

Q&A

- Open floor for questions.
- Discuss real-world applications of principles.