AGILE 2 MARKS

UNIT – IV

1. Agile Design Practices

Q: What are Agile design practices?
A: Agile design practices are principles that promote iterative, flexible, and customer-focused software development. They emphasize constant feedback, collaboration, and delivering working software in small, frequent releases.

2. Role of Design Principles

Q: What is the role of design principles in Agile?
A: Design principles in Agile ensure that the software is scalable, maintainable, and user-friendly. They guide developers to create simple, flexible, and adaptable solutions that can evolve as project requirements change.

3. Need and Significance of Refactoring

Q: Why is refactoring important in Agile?
A: Refactoring improves the internal structure of code without changing its external behavior, enhancing code readability, performance, and maintainability. It allows developers to accommodate new features with minimal risk of introducing errors.

4. Refactoring Techniques

Q: What are some common refactoring techniques?
A: Common refactoring techniques include renaming variables or methods for clarity, breaking large functions into smaller ones, eliminating duplicate code, and improving class structures. These techniques improve code quality and readability.

5. Continuous Integration

Q: What is continuous integration in Agile?
A: Continuous integration (CI) is the practice of regularly integrating code into a shared repository multiple times a day. This helps detect integration issues early, ensuring that the software remains functional and reducing the risk of defects.

6. Automated Build Tools

Q: What role do automated build tools play in Agile?
A: Automated build tools, like Jenkins or Maven, help automate the process of compiling, testing, and deploying code. They save time, reduce human errors, and ensure that the build process is consistent, fast, and reliable.

7. Version Control

Q: Why is version control important in Agile?
A: Version control systems like Git allow multiple developers to work on the same project without overwriting each other's work. It helps manage changes, track history, and roll back to previous versions when needed.

8. Agile Interaction Design

Q: What is Agile interaction design?
A: Agile interaction design involves creating and improving user interfaces iteratively, focusing on user feedback, collaboration, and flexibility. It ensures that user needs are met through constant refinement during the development process.

9. Agile Approach to Quality Assurance

Q: How does Agile approach quality assurance?
A: In Agile, quality assurance is integrated into the development process through continuous testing, collaboration between developers and testers, and frequent iterations. It emphasizes early detection and resolution of defects.

10. Test-Driven Development (TDD)

Q: What is Test-Driven Development (TDD)?
A: TDD is a development approach where tests are written before the code. Developers write a test for a specific function, then write the code to pass the test, ensuring the software works as expected and minimizing defects.

11. Pair Programming

Q: What is pair programming in Agile?
A: Pair programming is a technique where two developers work together on the same code. One writes the code (driver), while the other reviews and suggests improvements (observer). This leads to higher-quality code and knowledge sharing.

12. Issues and Challenges in Agile

Q: What are some challenges faced in Agile development?
A: Common challenges in Agile include managing stakeholder expectations, ensuring effective communication, maintaining team collaboration, dealing with scope creep, and managing iterative changes in requirements.