Software Testing
Assignment 5

Q1. Explain the key differences between Agile and Traditional SDLC models.

Aspect	Agile Model	Traditional (Waterfall/V-Model)
Process	Iterative & incremental	Linear & sequential
Flexibility	Highly flexible, changes allowed anytime	Rigid, changes difficult after design
Requirements	Evolving, gathered continuously	Fixed at the beginning
Customer Involvement	Continuous throughout project	Limited to initial and final stages
Delivery	Frequent working releases	Single final delivery
Documentation	Minimal, as needed	Heavy documentation
Testing	Continuous & parallel to development	Occurs only after development phase
Risk Management	Early detection due to iterations	Risks discovered late

Conclusion: Agile adapts to change and focuses on rapid delivery, while Traditional SDLC emphasizes detailed planning and strict phases.

Q2. Describe the Scrum process flow and explain the significance of the Product Backlog.

Scrum Process Flow

1. **Product Backlog Creation**
   All project requirements listed and prioritized.

2. **Sprint Planning**
   Team selects user stories for the upcoming sprint.

3. **Sprint Execution (2–4 weeks)**
   Development + testing activities.

4. **Daily Scrum (Stand-up meeting)**
   15-minute meeting for progress tracking.

5. **Sprint Review**
   Team demonstrates the completed product increment.

6. **Sprint Retrospective**
   Team reflects on improvements for next sprint.

7. **Increment Delivery**
   A potentially shippable product is released.

Significance of Product Backlog

- Contains all features, user stories, enhancements, and bug fixes.
- Prioritized by the **Product Owner**.
- Acts as the **single source of truth** for requirements.
- Drives sprint planning and team workload.
- Continuously refined to adapt to changing needs.

Q3. What are Agile Metrics? Discuss any two and their importance.

Agile Metrics are **quantitative measures** used to evaluate team performance, project health, and delivery efficiency in Agile projects.

1. Velocity

- Measures the amount of work (story points) completed per sprint.
- Helps in sprint planning and forecasting.
- Shows team productivity trends.

2. Burndown Chart

- Shows remaining work in a sprint.
- Helps track progress daily.
- Identifies delays or bottlenecks early.

Importance

- Improves predictability and planning
- Helps stakeholders track progress
- Enhances team accountability and performance

Q4. Explain the principles of the Agile Model and discuss its advantages and disadvantages compared to traditional approaches.

Principles of Agile (based on Agile Manifesto)

1. Customer satisfaction through early and continuous delivery.
2. Welcome changing requirements.
3. Deliver working software frequently.
4. Collaboration between business & development teams.
5. Build projects around motivated individuals.
6. Face-to-face communication is most effective.
7. Working software is primary measure of progress.
8. Sustainable development pace.
9. Technical excellence & good design improves agility.

10. Simplicity.
11. Self-organizing teams.
12. Regular reflection and improvement.

Advantages

- Highly flexible
- Faster delivery
- Better risk management
- High customer involvement
- Continuous quality improvement

Disadvantages

- Less predictable
- Requires skilled teams
- Incomplete documentation
- Difficult in large, fixed-budget projects
- Customer availability is essential

Q5. Key practices of eXtreme Programming (XP) and how they contribute to quality

Key XP Practices

1. **Pair Programming** – Improves code quality through peer review.
2. **Test-Driven Development (TDD)** – Write tests before code, ensures accuracy.
3. **Continuous Integration** – Frequent code integration reduces conflicts.

4. **Refactoring** – Improves structure without changing functionality.
5. **Small Releases** – Frequent delivery ensures customer feedback.
6. **Collective Code Ownership** – Everyone can modify any part of the code.
7. **Simple Design** – Avoid complexity, improves maintainability.
8. **Sustainable Pace** – Prevents burnout, ensures productivity.

Contribution to Quality & Collaboration

- Early detection of defects
- High-quality code due to TDD and pair programming
- Better teamwork due to shared ownership
- Faster response to changes
- Clean, maintainable architecture

Q6. Compare and contrast Kanban and Scrum methodologies.

Aspect	Scrum	Kanban
Structure	Fixed-length sprints	Continuous flow
Roles	Defined roles (PO, SM, Team)	No mandatory roles
Planning	Sprint-based planning	Continuous planning
Work Limits	Sprint goals	WIP (Work-in-progress) limits
Meetings	Daily Scrum, Review, Retrospective	Optional meetings
Flexibility	Less flexible during sprint	Highly flexible anytime

When to Use Scrum

- Projects needing structured iterations
- Teams preferring defined roles and events
- When requirements evolve gradually

When to Use Kanban

- Support or maintenance projects
- Continuous delivery environments
- Highly unpredictable workloads

Q7. Phases of the Agile Model and how they ensure flexibility

1. Concept / Requirement Gathering

- High-level user stories created.
- Requirements evolve continuously.

2. Iteration / Development

- Development + testing occurs in cycles.
- Frequent feedback incorporated.

3. Release

- Working software delivered to users.

4. Maintenance

- Bugs fixed and enhancements added.

Ensuring Flexibility & Continuous Improvement

- Frequent iterations encourage constant change.
- Regular retrospectives drive improvement.
- Customer feedback integrated every cycle.
- Teams adapt processes dynamically.

Q8. Agile Testing methods and their significance

Agile Testing Methods

1. Test-Driven Development (TDD)
2. Behaviour-Driven Development (BDD)
3. Continuous Integration Testing
4. Exploratory Testing
5. Acceptance Test-Driven Development (ATDD)

Significance

- Testing happens continuously, not at the end.
- Defects found early → lower cost.
- Encourages collaboration between testers, developers, and customers.
- Ensures working software at every iteration.
- Focuses on user requirements and frequent verification.

How Agile Testing differs from Traditional

Traditional Testing	Agile Testing
Happens after development	Happens during each iteration
Heavy documentation	Lightweight, fast feedback
Fixed requirements	Requirements evolve
Separate QA team	Cross-functional team

Q9. Agile frameworks: Crystal, DSDM, FDD, Lean — features & applications

1. Crystal Methodologies

- Family of lightweight Agile methods (Crystal Clear, Yellow, Orange).
- Focus: People, communication, and collaboration.
- Suitable for small-to-medium teams.

2. DSDM (Dynamic System Development Method)

- Focuses on full project lifecycle.
- Prioritizes features using MoSCoW (Must, Should, Could, Won't).
- Good for fixed deadlines and budgets.

3. FDD (Feature-Driven Development)

- Model-driven, feature-based development.
- Five steps: develop model → build feature list → plan → design → build.
- Suitable for large-scale and complex systems.

4. Lean Software Development

- Eliminates waste
- Optimizes workflow
- Fast delivery
- Focus on value-driven development
- Inspired by lean manufacturing

Applications

- Crystal → small teams needing flexibility
- DSDM → strict timelines and business-driven projects
- FDD → large systems needing structured feature planning
- Lean → efficiency-focused startups and enterprises

Q10. What is Kanban? How to implement it and Agile Metrics used?

Kanban in Agile Development

Kanban is a **visual workflow management system** using the Kanban board to limit work and optimize flow.

Implementing Kanban

1. **Visualize Workflow** using a board (To Do → In Progress → Done).
2. **Set Work-in-Progress (WIP) Limits** to prevent overload.
3. **Measure and optimize flow** (cycle time, lead time).
4. **Continuous delivery** without fixed iterations.
5. **Improve processes** via regular review of bottlenecks.

Key Agile Metrics in Kanban

1. **Lead Time** – Total time taken from task creation to completion.
2. **Cycle Time** – Time taken from when work starts until it is done.
3. **WIP Limit** – Maximum number of tasks allowed at once.
4. **Throughput** – Number of tasks completed in a given time.

Importance

- Improves team productivity
- Enhances predictability
- Reduces bottlenecks
- Accelerates continuous delivery