**Introduction to SDLC (Software Development Life Cycle)**

**Definition:**  
The SDLC is a systematic process used by software organizations to plan, design, develop, test, deploy, and maintain software applications.

**Phases of SDLC:**

1. **Requirement Analysis** – Understanding what the client or end-user needs.
2. **System Design** – Creating architecture, data models, and design documents.
3. **Implementation (Coding)** – Writing the actual code following the design.
4. **Testing** – Verifying that the software works correctly and meets requirements.
5. **Deployment** – Delivering the application to the end-users or production.
6. **Maintenance** – Fixing issues and making improvements post-deployment.

**SDLC Models:**

* **Waterfall Model:** Sequential, rigid process.
* **V-Model:** Emphasizes verification and validation.
* **Incremental Model:** Builds software in small functional parts.
* **Iterative Model:** Refines software through repeated cycles.
* **Spiral Model:** Focuses on risk analysis and prototyping.
* **Agile Model:** Emphasizes flexibility and iterative delivery.

**Goal:** Deliver high-quality software efficiently and predictably.

**Agile**

**Definition:**  
Agile is a **flexible, iterative, and collaborative** approach to software development that focuses on **customer satisfaction** and **continuous improvement**.

**Agile Principles (from the Agile Manifesto):**

* Individuals and interactions over processes and tools.
* Working software over comprehensive documentation.
* Customer collaboration over contract negotiation.
* Responding to change over following a plan.

**Agile Frameworks:**

* **Scrum** – Time-boxed iterations called sprints.
* **Kanban** – Visual workflow emphasizing continuous delivery.
* **XP (Extreme Programming)** – Focus on technical excellence and frequent releases.
* **Lean** – Reducing waste and optimizing flow.

**Key Features:**

* Iterative development.
* Regular feedback loops.
* Cross-functional teams.
* Transparency through ceremonies and boards.

| **Role** | **Responsibility** |
| --- | --- |
| **Product Owner** | **Defines product vision, manages backlog, and prioritizes work.** |
| **Scrum Master** | **Facilitates the process, removes blockers, and ensures Agile principles are followed.** |
| **Development Team** | **Cross-functional members who design, build, and test the product.** |

**Benefits of Agile**

**✅ Faster delivery and time to market  
✅ Better product quality through continuous testing  
✅ High customer satisfaction due to regular feedback  
✅ Improved team morale and ownership  
✅ Flexibility to adapt to changes  
✅ Continuous improvement through retrospectives**

**Example Scenario**

**Suppose your team is developing an E-commerce website:**

1. **Create a product backlog (Login, Product Search, Add to Cart, Checkout, etc.).**
2. **Pick a few stories for the first 2-week sprint. (Task1 –1 week)**
3. **Deliver a working Login & Product Search feature at sprint end. 2week -submit**
4. **Gather customer feedback, refine backlog, and start the next sprint.**
5. **Continue this cycle until the full product is ready.**

**3. Project Boards**

**Definition:**  
Project boards (like **JIRA**, **Trello**, or **Azure Boards**) visualize and track the progress of work in Agile projects.

**Types of Boards:**

1. **Scrum Board:** Tracks sprint progress — tasks move from “To Do” → “In Progress” → “Done”.
2. **Kanban Board:** Continuous flow — no fixed sprints, focuses on limiting work in progress (WIP).

**Columns Typically Include:**

* **Backlog:** Upcoming tasks/features.
* **To Do:** Selected tasks for the sprint.
* **In Progress:** Currently active tasks.
* **Testing/Review:** Awaiting validation.
* **Done:** Completed work.

**Benefits:**

* Transparency of team activities.
* Quick identification of bottlenecks.
* Easy tracking of sprint progress.

**4. Story Pointing and Burndown Chart**

**Story Pointing**

**Definition:**  
A **relative estimation technique** used to measure the **effort and complexity** of user stories.

**Units:** Story points (not hours) – they consider:

* Complexity
* Risk
* Amount of work

**Estimation Methods:**

* **Planning Poker:** Team members assign story points collaboratively.
* **Fibonacci Series (1, 2, 3, 5, 8, 13...)** – commonly used scale.

Scales Used for Story Points

Most teams use a Fibonacci-like sequence since estimation accuracy drops for large numbers:

1, 2, 3, 5, 8, 13, 20, 40, 100

* 1 Point: Very simple, small change.
* 3 Points: Medium complexity, some testing needed.
* 8 Points: Large or complex feature.
* 13+ Points: Too big — should be broken into smaller stories.

**Purpose:**  
Helps teams predict how much work can be completed in a sprint (velocity).

**Burndown Chart**

**Definition:**  
A **graphical representation** of remaining work versus time in a sprint.

Pending task ,remaining time ----burndowndown –(Countdown)

**Axes:**

* **X-axis:** Time (days in sprint)
* **Y-axis:** Remaining story points

**Interpretation:**

* A smooth downward line → sprint is on track.
* Flat line → work not progressing.
* Sudden drops → bulk updates near sprint end (not ideal).

**Use:**  
Helps Scrum Masters and teams monitor progress and forecast completion.

**5. Scrum Ceremonies**

Scrum defines **five main ceremonies (meetings)** to ensure collaboration and inspection.

| **Ceremony** | **Purpose** | **Frequency** | **Participants** |
| --- | --- | --- | --- |
| **Sprint Planning** | Define sprint goal and select backlog items | Start of each sprint | Scrum Team |
| **Daily Stand-up (Daily Scrum)** | 15-min sync on progress, blockers, next steps | Daily | Development Team |
| **Sprint Review** | Demonstrate completed work to stakeholders | End of sprint | Scrum Team + Stakeholders |
| **Sprint Retrospective** | Reflect on process improvements | After Sprint Review | Scrum Team |
| **Backlog Refinement (Grooming)** | Update and prioritize product backlog | Ongoing (weekly) | Product Owner + Team |

**Goal:**  
Ensure **transparency, alignment, and continuous improvement**.