**Software development life cycle (SDLC)**

**Software development life cycle (SDLC) is a structured process that is used to design, develop, and test good-quality software.** SDLC, or software development life cycle, is a methodology that defines the entire procedure of software development step-by-step.

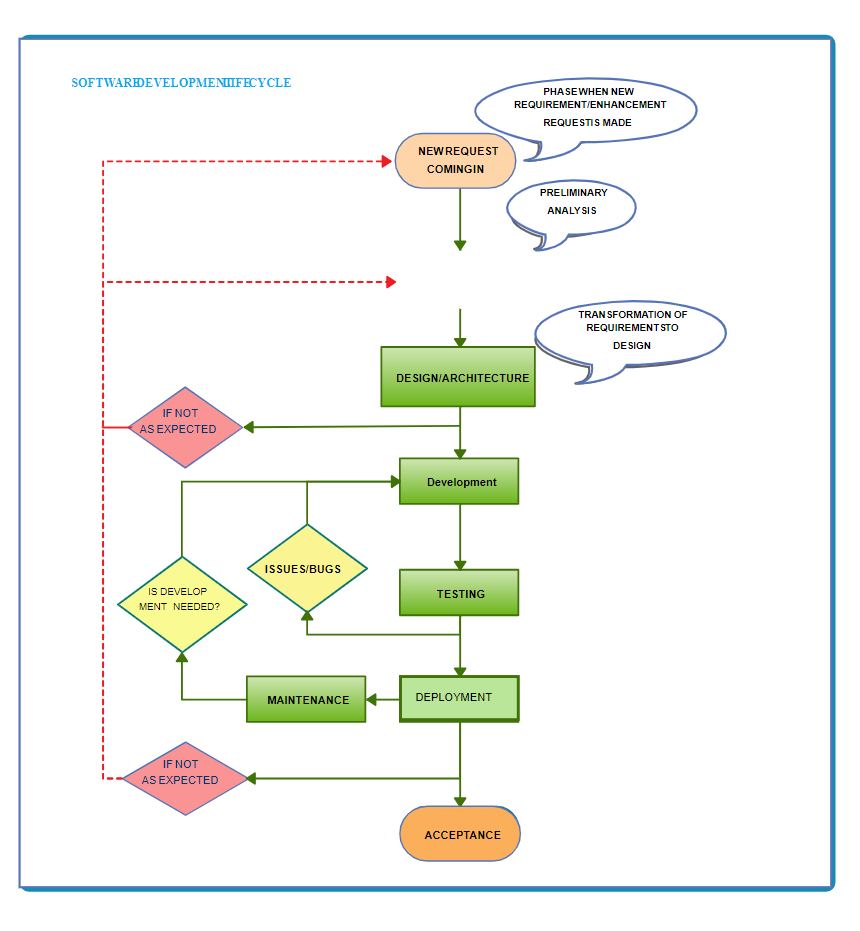


*Software Development Life Cycle (SDLC)*

**Phases of SDLC:**

* Phase 1: Requirements Analysis.
* Phase 2: Design.
* Phase 3: Implementation
* Phase 4: Testing.
* Phase 5: Deployment.
* Phase 6: Maintenance.

**Flow chart of the SDLC:**



In the Software Development Life Cycle (SDLC), each phase involves specific activities:

1. Planning: Define the project scope and objectives.

2. Requirement Analysis Gather and document functional and non-functional requirements.

3. Design: Create system architecture and design specifications.

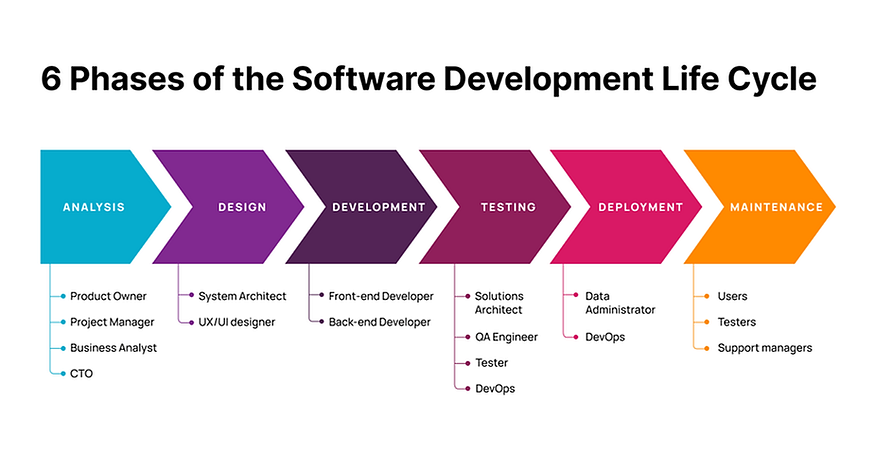
4. Implementation (Coding): Write and compile the source code.

5. Testing: Verify and validate the software through various tests.

6. Deployment: Install and configure the software in the production environment.

7. Maintenance: Perform ongoing support and updates to the software.

**Who are invloved in each phase**



**What is the outcome from each phase**

In the Software Development Life Cycle (SDLC), each phase produces specific deliverables or outcomes:

1. Requirement Gathering: Requirements specification document.

2. Design: System and software design documents.

3. Implementation (Coding): Source code and compiled software.

4. Testing: Test reports and defect logs.

5. Deployment: Deployed application and deployment documentation.

6.Maintenance: Updated software and maintenance logs.

**SDLC MODELS**

Software Development life cycle (SDLC) is a spiritual model used in project management that defines the stages include in an information system development project, from an initial feasibility study to the maintenance of the completed application.

There are different software development life cycle models specify and design, which are followed during the software development phase. These models are also called "**Software Development Process Models**." Each process model follows a series of phase unique to its type to ensure success in the step of software development.

1.Waterfall Model

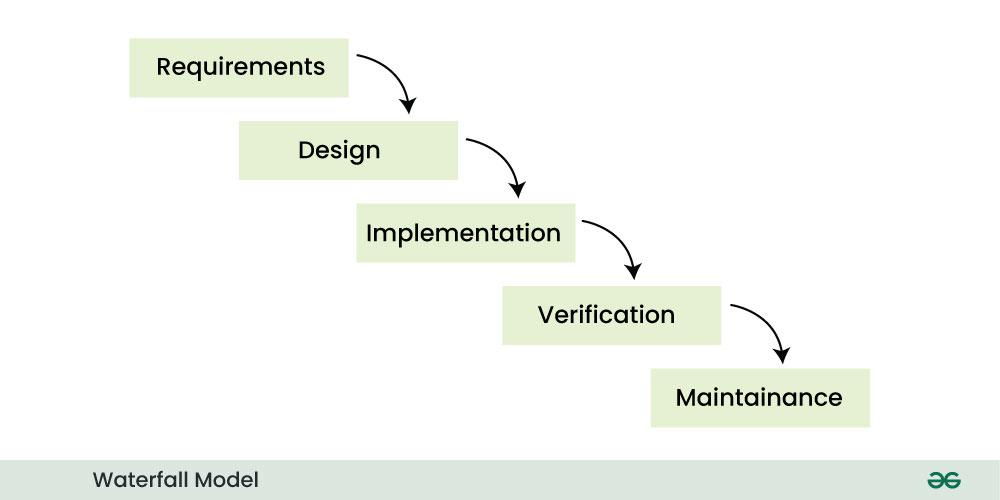
2.V Model

3.Spiral Model

4.Agile Model

1.Waterfall Model:

The waterfall model is a continuous software development model in which development is seen as flowing steadily **downwards** (like a waterfall) through the steps of requirements analysis, design, implementation, testing (validation), integration, and maintenance.



**Advantages of Waterfall Model**

* This model is simple and easy to understand.
* This is very useful for small projects.
* This model is easy to manage.
* The end goal is determined early.
* Each phase of this model is well explained.
* It provides a structured way to do things.
* This is a base model, all the SDLC models that came after this were created keeping this in mind, although they worked to remove its shortcomings.
* In this model, we can move to the next phase only after the first phase is successfully completed so that there is no overlapping between the phases.

**Disadvantages of Waterfall Model**

* In this model, complete and accurate requirements are expected at the beginning of the development process.
* Working software is not available for very long during the development life cycle.
* We cannot go back to the previous phase due to which it is very difficult to change the requirements.
* Risk is not assessed in this, hence there is high risk and uncertainty in this model.
* In this the testing period comes very late.
* Due to its sequential nature this model is not realistic in today’s world.
* This is not a good model for large and complex projects.

**2. V-Model**

V-Model is an SDLC model, it is also called Verification and Validation Model.  In V-Model, the execution of each process is sequential, that is, the new phase starts only after the previous phase ends.

* It is based on the association of testing phase with each development phase that is in V-Model with each development phase, its testing phase is also associated in a V-shape in other words both [software development](https://www.geeksforgeeks.org/software-development/) and testing activities take place at the same time.
* So in this model, Verification Phase will be on one side, Validation Phase will be on the other side that is both the activities run simultaneously and both of them are connected to each other in V-Shape through Coding Phase, hence it is called V-Model.



**Advantages of V-Model**

* This is a simple and easy to use model.
* Planning, testing and designing tests can be done even before coding.
* This is a very disciplined model, in which phase by phase development and testing goes on.
* Defects are detected in the initial stage itself.
* Small and medium scale developments can be easily completed using it.

**Disadvantages of V-Model**

* This model is not suitable for any complex projects.
* There remains both high risk and uncertainty.
* This is not a suitable model for an ongoing project.
* This model is not at all suitable for a project which is unclear and in which there are changes in the requirement.

**3.Spiral Model**

Spiral model is a [software development process](https://www.geeksforgeeks.org/software-development-process/) model. This model has characteristics of both iterative and waterfall models. This model is used in projects which are large and complex. A software project goes through these loops again and again in iterations. After each iteration a more and more complete version of the software is developed. The most special thing about this model is that risks are identified in each phase and they are resolved through prototyping. This feature is also called Risk Handling.



**Advantages of Spiral Model**

* If we have to add additional functionality or make any changes to the software, then through this model we can do so in the later stages also.
* Spiral model is suitable for large and complex projects.
* It is easy to estimate how much the project will cost.
* Risk analysis is done in each phase of this model.
* The customer can see the look of his software only in the early stages of the development process.
* Since continuous feedback is taken from the customer during the development process, the chances of customer satisfaction increases.

**Disadvantage of Spiral Model**

* This is the most complex model of SDLC, due to which it is quite difficult to manage.
* This model is not suitable for small projects.
* The cost of this model is quite high.
* It requires more documentation than other models.
* Experienced experts are required to evaluate and review the project from time to time.
* Using this model, the success of the project depends greatly on the risk analysis phase.

**4.Agile Model**

Agile model is a combination of iterative and incremental models, that is, it is made up of iterative and incremental models.

* In Agile model, focus is given to process adaptability and customer satisfaction.
* In earlier times, iterative waterfall model was used to create software. But in today’s time developers have to face many problems. The biggest problem is that in the middle of software development, the customer asks to make changes in the software. It takes a lot of time and money to make these changes.



* In the agile model, the software product is divided into small incremental parts. In this, the smallest part is developed first and then the larger one.
* And each incremental part is developed over iteration.
* Each iteration is kept small so that it can be easily managed. And it can be completed in two-three weeks. Only one iteration is planned, developed and deployed at a time.

**Advantages of Agile Model**

* In this, two programmers work together due to which the code is error free and there are very few mistakes in it.
* In this the software project is completed in a very short time.
* In this the customer representative has an idea of ​​each iteration so that he can easily change the requirement.
* This is a very realistic approach to software development.
* In this, focus is given on teamwork.
* There are very few rules in this and documentation is also negligible.
* There is no need for planning in this.
* It can be managed easily.
* It provides flexibility to developers.

**Disadvantages of Agile Model**

* It cannot handle complex dependencies.
* Due to lack of formal documentation in this, there is confusion in development.
* It mostly depends on the customer representative, if the customer representative gives any wrong information then the software can become wrong.
* Only experienced programmers can take any decision in this. New programmers cannot take any decision.
* In the beginning of software development, it is not known how much effort and time will be required to create the software.

Test-Driven Development (TDD)

Behavior-Driven Development (BDD),

Feature-Driven Development (FDD)

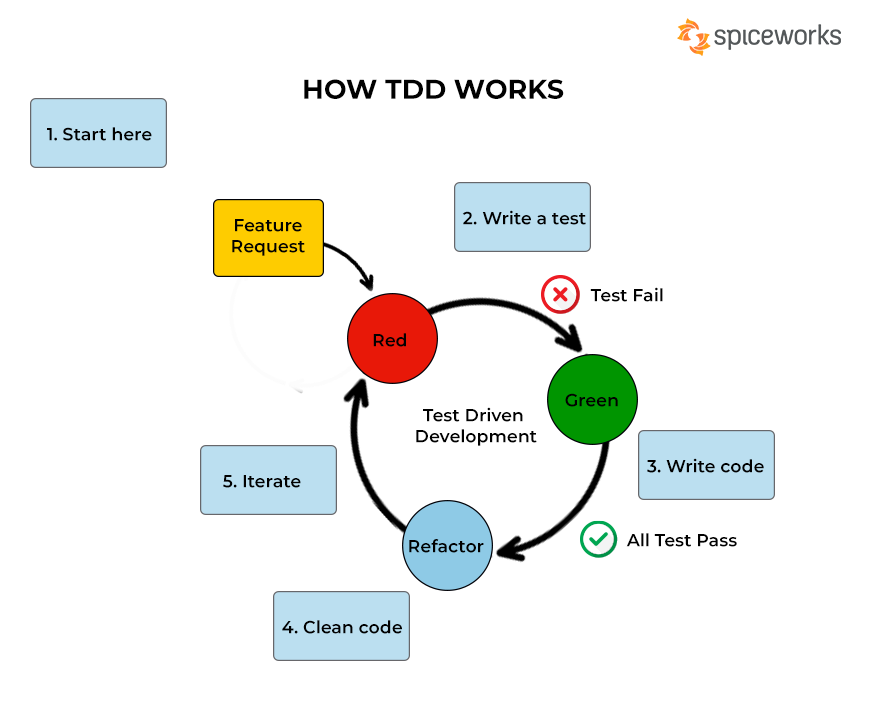
**1. Test-Driven Development (TDD)**

**Concept:**  
TDD is a software development approach where tests are written before the actual code. The process involves writing a test for a small piece of functionality, writing the minimal code required to pass the test, and then refactoring the code.

**Process:**

1. **Write a Test:** Define a test for a new feature or functionality.
2. **Run the Test:** Check if the new test fails (since the feature isn't implemented yet).
3. **Write Code:** Implement the minimal amount of code to pass the test.
4. **Run Tests:** Ensure all tests pass.
5. **Refactor Code:** Improve the code without changing its behavior.
6. **Repeat:** Continue the cycle for additional features.

**Diagram**



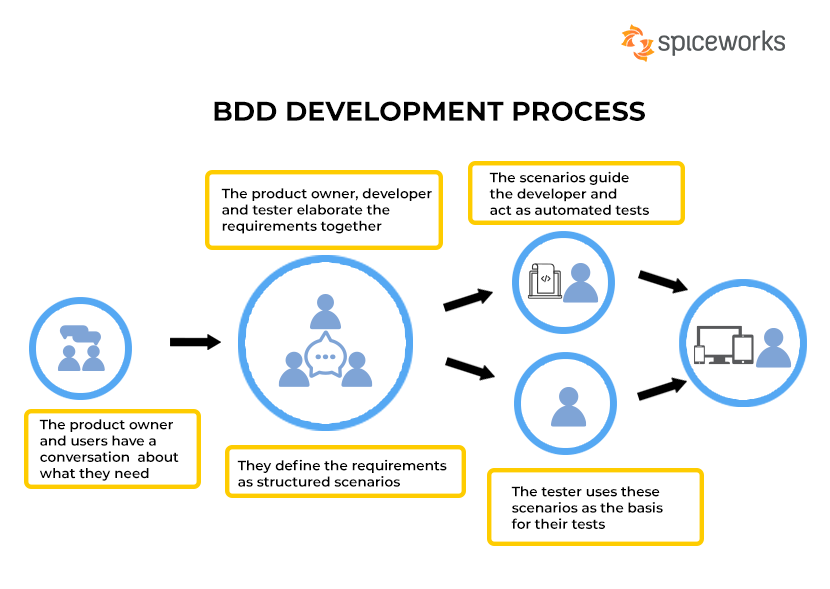
**2. Behavior-Driven Development (BDD)**

**Concept:**  
BDD is an extension of TDD that focuses on the behavior of the application as perceived by the end user. It encourages collaboration between developers, QA, and non-technical stakeholders by using a common language (often English-like syntax) to describe the behavior of the system.

**Process:**

1. **Define Scenarios:** Write scenarios in a format like Given-When-Then, describing expected behavior.
2. **Automate Scenarios:** Implement automated tests for these scenarios.
3. **Develop Feature:** Write code to make the scenarios pass.
4. **Run Scenarios:** Validate that the scenarios pass.
5. **Refactor:** Improve the implementation without changing the behavior.

**Diagram:**



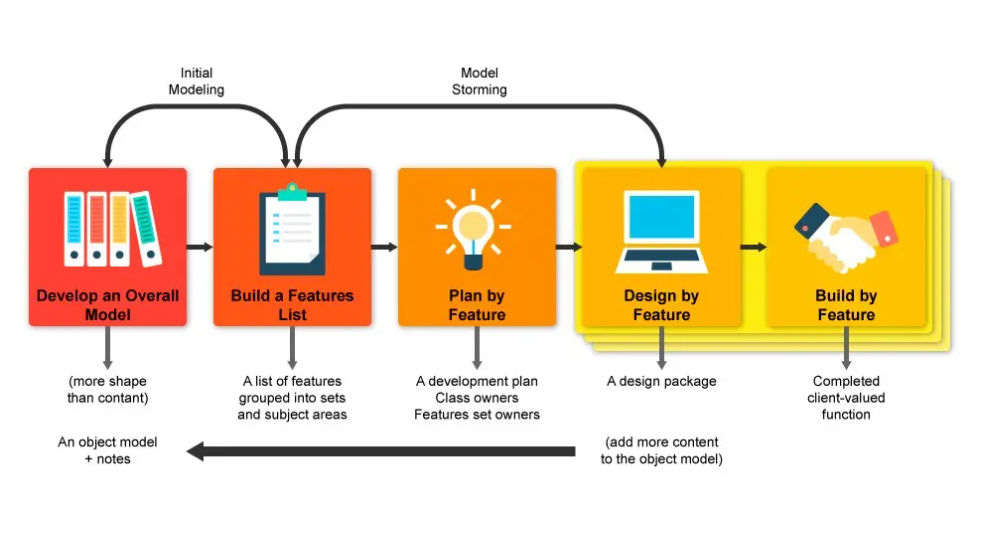
**3. Feature-Driven Development (FDD)**

**Concept:**  
FDD is a model-driven, short-iteration process. It involves defining and building features—small, client-valued functions that can be delivered in weeks rather than months. FDD focuses on an incremental approach, providing specific, short-term value to clients.

**Process:**

1. **Develop Overall Model:** Build a high-level model of the domain.
2. **Build Feature List:** Identify features based on client needs.
3. **Plan by Feature:** Prioritize and plan features.
4. **Design by Feature:** Detailed design for each feature.
5. **Build by Feature:** Implement, test, and review each feature.

**Diagram:**



1. **Test-Driven Development (TDD)**:
   * Focuses on writing tests before the actual code.
   * Ensures code correctness and supports continuous integration.
2. **Behavior-Driven Development (BDD)**:
   * Emphasizes collaboration between developers and non-technical stakeholders.
   * Uses scenarios written in natural language to describe system behavior.
3. **Feature-Driven Development (FDD)**:
   * A model-driven approach that focuses on features.
   * Breaks down the project into small, manageable parts, delivering features iteratively.

**Agile ceremonies:**

Agile ceremonies are structured meetings or events that occur regularly in Agile project management methodologies. They are designed to ensure consistent communication, collaboration, and progress within the team. Here are the key Agile ceremonies:

**1. Sprint Planning**

**Purpose:**

To define the work that will be completed in the upcoming sprint.

**Activities:**

- Review the product backlog and prioritize items.

- Define the sprint goal and select the backlog items to work on.

- Break down selected items into tasks.

- Estimate the effort required for each task.

**Participants:**

Product Owner, Scrum Master, Development Team.

**Frequency:**

At the beginning of each sprint.

**2. Daily Stand-up (Daily Scrum):**

**Purpose:**

To synchronize the team and plan the day's work.

**Activities:**

- Each team member answers three questions:

1. What did I accomplish yesterday?

2. What will I work on today?

3. Are there any blockers or impediments?

**Participants:**

Development Team, Scrum Master (Product Owner can attend but is not mandatory).

**Frequency:**

Daily, typically time-boxed to 15 minutes.

**3. Sprint Review**

**Purpose:**

To demonstrate and inspect the work completed during the sprint.

**Activities:**

- Present the work done in the sprint to stakeholders.

- Gather feedback and discuss potential improvements.

- Review progress towards the project goals.

**Participants:**

Development Team, Scrum Master, Product Owner, Stakeholders.

**Frequency:**

At the end of each sprint.

4. **Sprint Retrospective**

**Purpose:**

To reflect on the sprint and identify ways to improve the process.

**Activities:**

- Discuss what went well during the sprint.

- Identify what didn't go well and what could be improved.

- Create action items for improvement in the next sprint.

**Participants**:

Development Team, Scrum Master, Product Owner.

**Frequency**:

At the end of each sprint, after the Sprint Review.

**5. Backlog Refinement (Grooming)**

**Purpose**:

To ensure the product backlog is up-to-date and ready for sprint planning.

**Activities**:

- Review and prioritize backlog items.

- Break down large items into smaller, more manageable tasks.

- Clarify requirements and estimate effort.

**Participants**:

Product Owner, Development Team (Scrum Master may facilitate).

**Frequency**:

Regularly throughout the sprint, often as an ongoing process.

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*requirements of the “zoom” Application\*\*\*\*\*\***

**Functionality requirements for a Zoom:**

1. **User Authentication**: Secure login and registration using email, social media, or SSO.

2. **User Profiles**: Manage personal information and settings.

3. **Video Conferencing**: Real-time video and audio communication with multiple participants.

4. **Screen Sharing**: Ability to share screen content with others during meetings.

5. **Meeting Scheduling**: Schedule, manage, and send calendar invites for meetings.

6**. Chat:** In-meeting text chat for participants.

7. **Recording:** Record meetings for later playback and download.

8. **Virtual Backgrounds:** Option to use virtual backgrounds during video calls.

9. **Meeting Controls:** Host controls for muting participants, managing attendees, and locking meetings.

10. **Breakout Rooms:** Create and manage smaller groups within a meeting.

11. **Waiting Room:** Pre-meeting space to manage attendee entry.

12. **File Sharing:** Upload and share files during meetings.

13. **Notifications:** Push notifications and reminders for upcoming meetings.

14. **Integration with Calendar Apps:** Sync with Google Calendar, Outlook, etc.

15. **Security Features**: End-to-end encryption, meeting passwords, and two-factor authentication.

**Update Functionality requirements for a Zoom:**

1. \*\*Enhanced Breakout Rooms\*\*: Add features for random assignment, room timer, and custom room naming.

2. \*\*Live Polling\*\*: Introduce live polling with real-time results sharing during meetings.

3. \*\*Virtual Background Customization\*\*: Allow users to customize and save multiple virtual backgrounds.

4. \*\*Meeting Analytics\*\*: Provide detailed analytics on meeting duration, participation, and engagement.

5. \*\*Integrated Whiteboard\*\*: Implement a collaborative whiteboard feature for brainstorming and drawing.

6. \*\*Transcription and Translation\*\*: Enable automatic transcription and real-time translation for different languages.

7. \*\*Increased Participant Limits\*\*: Expand the maximum number of participants allowed in a meeting.

8. \*\*Advanced Security Options\*\*: Introduce end-to-end encryption and two-factor authentication for added security.

9. \*\*Emoji Reactions\*\*: Allow participants to use emoji reactions during meetings without interrupting the speaker.

10. \*\*Custom Meeting Layouts\*\*: Provide options for customizing the layout of video feeds and shared content.

11. \*\*Offline Meeting Scheduling\*\*: Enable users to schedule meetings and set reminders offline, with automatic sync upon reconnection.

12. \*\*Integration with Calendar Apps\*\*: Integrate with major calendar apps for seamless meeting scheduling and reminders.

13. \*\*Multi-host Support\*\*: Allow multiple hosts in a single meeting with shared control over meeting settings.

14. \*\*File Sharing\*\*: Implement a secure file-sharing feature within the meeting interface.

15. \*\*Performance Optimization\*\*: Optimize the application for low-bandwidth connections and older devices.

Describe scrum and kanban. List out the scrum and kanban

**Scrum:**

Scrum is a framework for getting work done within agile. Scrum uses all the core principles of agile to define methods to facilitate a project**.**

**Key components of the Scrum Development Model:**

**1. Roles:**

* **Product Owner:** Represents the stakeholders and is responsible for defining and prioritizing the product backlog.
* **Scrum Master:** Facilitates the Scrum process and ensures that the team adheres to its practices.
* **Development Team:**Cross-functional and self-organizing group responsible for delivering the product increment.

**2. Artifacts:**

* **Product Backlog:** A prioritized list of features, enhancements, and bug fixes that need to be addressed in the product.
* **Backlog:** A subset of the product backlog selected for a specific sprint, containing tasks the team commits to completing.
* **Product Increment:**The sum of all the completed product backlog items at the end of a sprint.

**3. Events:**

* **Sprint:** A time-boxed iteration (usually 2-4 weeks) during which a potentially shippable product increment is created.
* **Sprint Planning**: A meeting at the beginning of each sprint where the team plans the work to be done.
* **Daily Scrum (Stand-up):** A short daily meeting where team members discuss progress, plan for the day, and identify and address impediments.
* **Sprint Review:** A meeting at the end of each sprint to review the completed work and gather feedback.
* **Sprint Retrospective:** A meeting at the end of each sprint for the team to reflect on their processes and identify improvements.

**4. Rules:**

* Scrum emphasizes transparency, inspection, and adaptation.
* The product is built incrementally in fixed-length iterations (sprints).
* Changes are only made between sprints unless there is a compelling reason to make a change during a sprint.

Scrum provides a flexible and collaborative approach to project management, allowing teams to adapt to changing requirements and deliver a potentially shippable product at the end of each sprint. It is widely used in various industries for its focus on iterative development, continuous improvement, and customer feedback.

**Kanban:**

In software development, Kanban is used to control and efficiently manage the flow of features (represented by Kanban cards) under development. The stages in the SDLC are used to represent the different stages in the manufacturing process, and the number of features entering the process should match the completed ones.

* Tasks could remain in the ‘wait’ state for long.
* Mainly research-oriented takes are there.
* For enhancements where requirements are evolving/ unclear.
* No prior scope is not defined and tasks keep on evolving.
* Too much dependency is there between tasks.
* If all the items across work stages need to be collated, then only deployed.

Differences between scrum and Kanban:

| S. No. | Scrum | Kanban |
| --- | --- | --- |
| 1. | It defines the role of each member of the Scrum team. | There is no role assigned to individuals. |
| 2. | It follows the iterative method. | It does not follow the iterative approach |
| 3. | To solve a problem, it breaks it into small tasks and then processes it further. | It does not break a problem into sub-problems. |
| 4. | It is a highly prescriptive approach. | It is not much prescriptive as compared to Scrum. |
| 5. | There is no visualization process to perform tasks. | There is a visualization process to perform tasks. |
| 6. | There are sprints that keep track of the progress of any project. | They use task cards to keep track of the progress of any project. |
| 7. | It is processed in successive sprints to complete a task. | It is used to optimize the task to complete a project. |
| 8. | It is not preferred when resources are limited. | It is preferred when tasks and resources are limited. |
| 9. | Scrum Master is the problem solver in case of a problem. | All the members are allowed to pick a problem and solve it. |
| 10. | The process does not get disturbed if a team member leaves in between a sprint. | The flow of work gets disturbed if a team member leaves in between. |
| 11. | The velocity of the sprint is used to measure the production. | The time taken to finish the project is the measure of production. |
| 12. | Estimation is crucial to Scrum because it places a strong emphasis on planning. | Estimation is not as important in Kanban as in scrum. |
| 13. | In scrum, cross-functional teams are important to deal with the issues that may occur during software development. | In Kanban, specialized teams are important. |
| 14. | Only one team owns a sprint backlog. | The sharing among multiple teams is possible with the Kanban board. |
| 15. | The scrum methodology is centered on the backlog. | The Kanban methodology is centered on the process dashboard. |
| 16. | It is suitable for projects that have changing priorities. | It is suitable for projects that have stable priorities i.e. unlikely to change over time. |
| 17. | “Velocity” through sprints is a production measurement metric. | “Cycle time” is a production measurement metric. |
| 18. | One to four weeks make up a sprint cycle. | The delivery cycle is continuous. |
| 19. | Some of the Tools-   * Jira Software * Axosoft * VivifyScrum and more. | Some of the Tools-   * Jira Software * Kanbanize * SwiftKanban * Asana and more. |

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

**Consequences of Not Testing Software:**

1. **Undetected Bugs:** Software may contain hidden defects that can cause malfunctions, crashes, or incorrect results, leading to user frustration and dissatisfaction.
2. **Security Vulnerabilities:** Without proper testing, security flaws may go unnoticed, exposing the software to hacking, data breaches, and other security threats.
3. **Poor Quality:** The software may fail to meet quality standards, resulting in functionality issues, poor performance, and a lack of necessary features.
4. **Higher Costs:** Fixing bugs after release is more costly and time-consuming than addressing them during development. It can also lead to financial losses due to refunds, legal issues, or loss of customers.
5. **Damage to Reputation:** Releasing a flawed product can damage the reputation of the company, leading to loss of trust and customer loyalty.
6. **Project Failures:** In severe cases, critical defects in the software can lead to project failures, especially in systems where safety and precision are paramount.