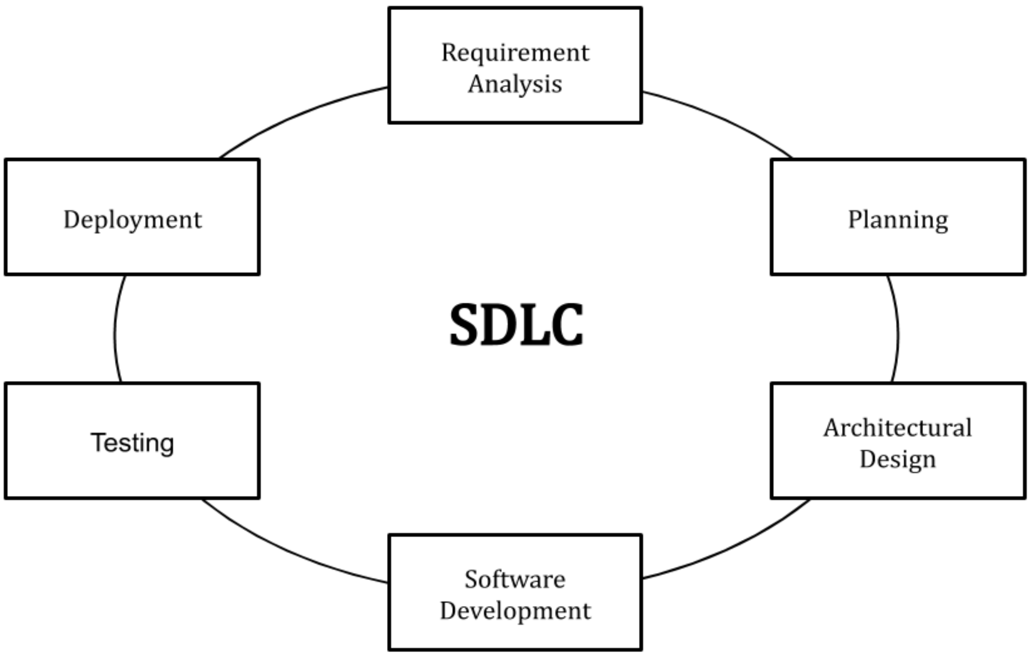
Software Development Life Cycle

➢ Explain SDLC at a high level ?

The Software Development Life Cycle (SDLC) refers to a methodology with clearly defined processes for creating high-quality software. **SDLC** is a systematic process for building software that ensures the quality and correctness of the software built. SDLC process aims to produce high-quality software that meets customer expectations. The system development should be complete in the pre-defined time frame and cost. SDLC consists of a detailed plan which explains how to plan, build, and maintain specific software. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase, the SDLC methodology focuses on the following phases of software development:

- Requirement analysis
- Planning
- Software design such as architectural design
- Software development
- Testing
- Deployment and Maintenance



SDLC works by lowering the cost of software development while simultaneously improving quality and shortening production time. SDLC achieves these apparently divergent goals by following a plan that removes the typical pitfalls of software development projects. That plan starts by evaluating existing systems for deficiencies.

Next, it defines the requirements of the new system. It then creates the software through the stages of analysis, planning, design, development, testing, and deployment. By anticipating costly mistakes like failing to ask the end-user or client for feedback, SLDC can eliminate redundant rework and after-the-fact fixes.

It’s also important to know that there is a strong focus on the testing phase. As the SDLC is a repetitive methodology, you have to ensure code quality at every cycle. Many organizations tend to spend few efforts on testing while a stronger focus on testing can save them a lot of rework, time, and money. Be smart and write the right types of tests.

Phase 1: Requirement collection and analysis:

The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and recognization of the risks involved is also done at this stage.

This stage gives a clearer picture of the scope of the entire project and the anticipated issues, opportunities, and directives which triggered the project.

Requirements Gathering stage need teams to get detailed and precise requirements. This helps companies to finalize the necessary timeline to finish the work of that system.

Phase 2: Feasibility study:

Once the requirement analysis phase is completed the next sdlc step is to define and document software needs. This process conducted with the help of 'Software Requirement Specification' document also known as 'SRS' document. It includes everything which should be designed and developed during the project life cycle.

There are mainly five types of feasibilities checks:

- **Economic:** Can we complete the project within the budget or not?
- **Legal:** Can we handle this project as cyber law and other regulatory framework/compliances.
- **Operation feasibility:** Can we create operations which is expected by the client?
- **Technical:** Need to check whether the current computer system can support the software
- **Schedule:** Decide that the project can be completed within the given schedule or not.

Phase 3: Design:

In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture.

This design phase serves as input for the next phase of the model.

There are two kinds of design documents developed in this phase:

High-Level Design (HLD)

- Brief description and name of each module
- An outline about the functionality of every module
- Interface relationship and dependencies between modules
- Database tables identified along with their key elements
- Complete architecture diagrams along with technology details

Low-Level Design(LLD)

- Functional logic of the modules
- Database tables, which include type and size
- Complete detail of the interface
- Addresses all types of dependency issues
- Listing of error messages
- Complete input and outputs for every module

Phase 4: Coding:

Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

In this phase, Developer needs to follow certain predefined coding guidelines. They also need to use programming tools like compiler, interpreters, debugger to generate and implement the code.

Phase 5: Testing:

Once the software is complete, and it is deployed in the testing environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

During this phase, QA and testing team may find some bugs/defects which they communicate to developers. The development team fixes the bug and send back to QA for a re-test. This process continues until the software is bug-free, stable, and working according to the business needs of that system.

Phase 6: Installation/Deployment:

Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

Phase 7: Maintenance:

Once the system is deployed, and customers start using the developed system, following 3 activities occur

- Bug fixing - bugs are reported because of some scenarios which are not tested at all

- Upgrade - Upgrading the application to the newer versions of the Software
- Enhancement - Adding some new features into the existing software

The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase.

➢ What is waterfall model and why it is still relevant

The waterfall is a widely accepted SDLC model. In this approach, the whole process of the software development is divided into various phases of SDLC. In this SDLC model, the outcome of one phase acts as the input for the next phase.

. Though many may see Waterfall as a slightly out-dated technology, many product developers stand by it. These are some benefits of the implementation of Waterfall methodologies, which include the following:

- It is a simple and quite straightforward approach
- It is easy to develop an overall plan for managing a Waterfall project because every phase has a specific start and end. This allows the team to ascertain exactly what coding is to be developed, when it’s due and when testing should begin. This can all be done before the commencement of the project
- Early planning will ensure a solid basis for designing components that can seamlessly integrate with external systems
- Planning resources for Waterfall is generally easier as you know exactly when everything will start and end
- Clients who prefer specific start and end dates will appreciate Waterfall as this model allows them to know the exact date when they will have their product in their hands
- The development costs can be determined before the project starts
- Extremely detailed procedures can be used effectively to regulate all parts of the project, preventing any errors from occurring during the building process
- A suitable solution for clients who prefer not to be involved in the development process and simply want to be involved in the beginning and then receive a complete product at the end
- Ideal for small projects, where the speed of delivery is not of utmost importance
- Situations where similar projects will be carried out in the future, allowing the project plan to be reused, making use of the heavy documentation that was drawn up previously

Software Development Life Cycle

➢ **Explain Agile Model with a use case and the role of SCRUM in that**

AGILE methodology is a practice that promotes **continuous iteration** of development and testing throughout the software development lifecycle of the project. In the Agile model, both development and testing activities are concurrent, unlike the Waterfall model.

Each iteration is considered as a short time "frame" in the Agile process model, which typically lasts from one to four weeks. The division of the entire project into smaller parts helps to minimize the project risk and to reduce the overall project delivery time requirements. Each iteration involves a team working through a full software development life cycle including planning, requirements analysis, design, coding, and testing before a working product is demonstrated to the client.

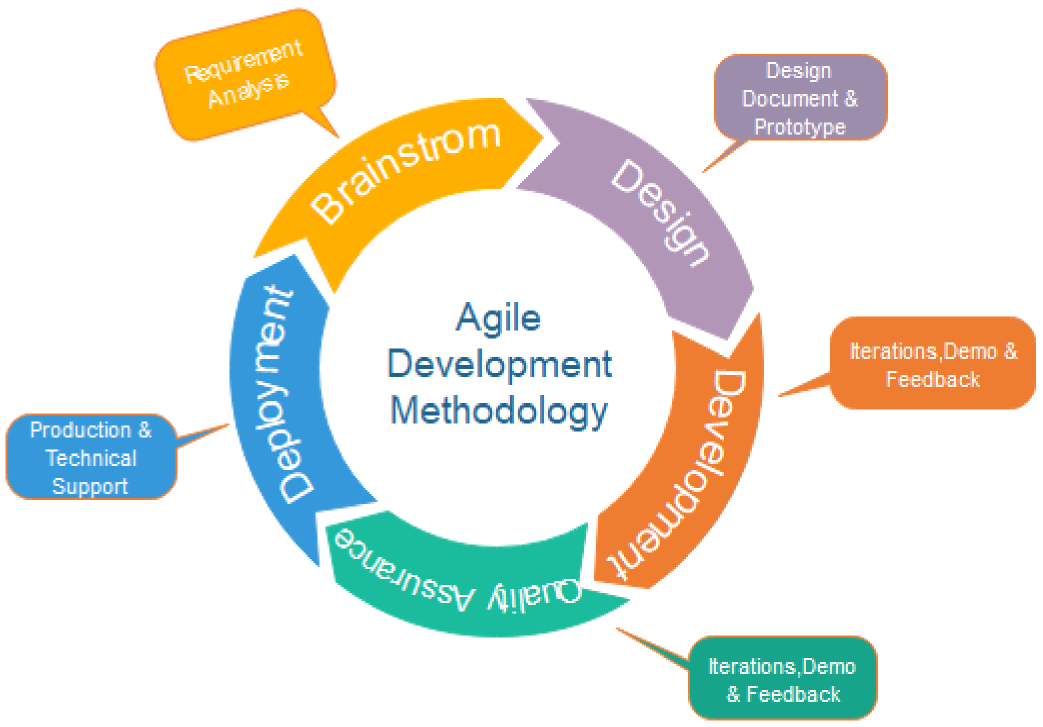

Requirement Analysis
Design Document & Prototype
Brainstrom
Design
Agile Development Methodology
Iterations,Demo & Feedback
Development
Production & Technical Support
Deployment
Quality Assurance
Iterations,Demo & Feedback

Fig. Agile Model

Example: Adobe is working on project to come up with a competing product for Microsoft Word, that provides all the features provided by Microsoft Word and any other features requested by the marketing team. The final product needs to be ready in 10 months of time. Let us see how this project is executed in traditional and Agile methodologies.

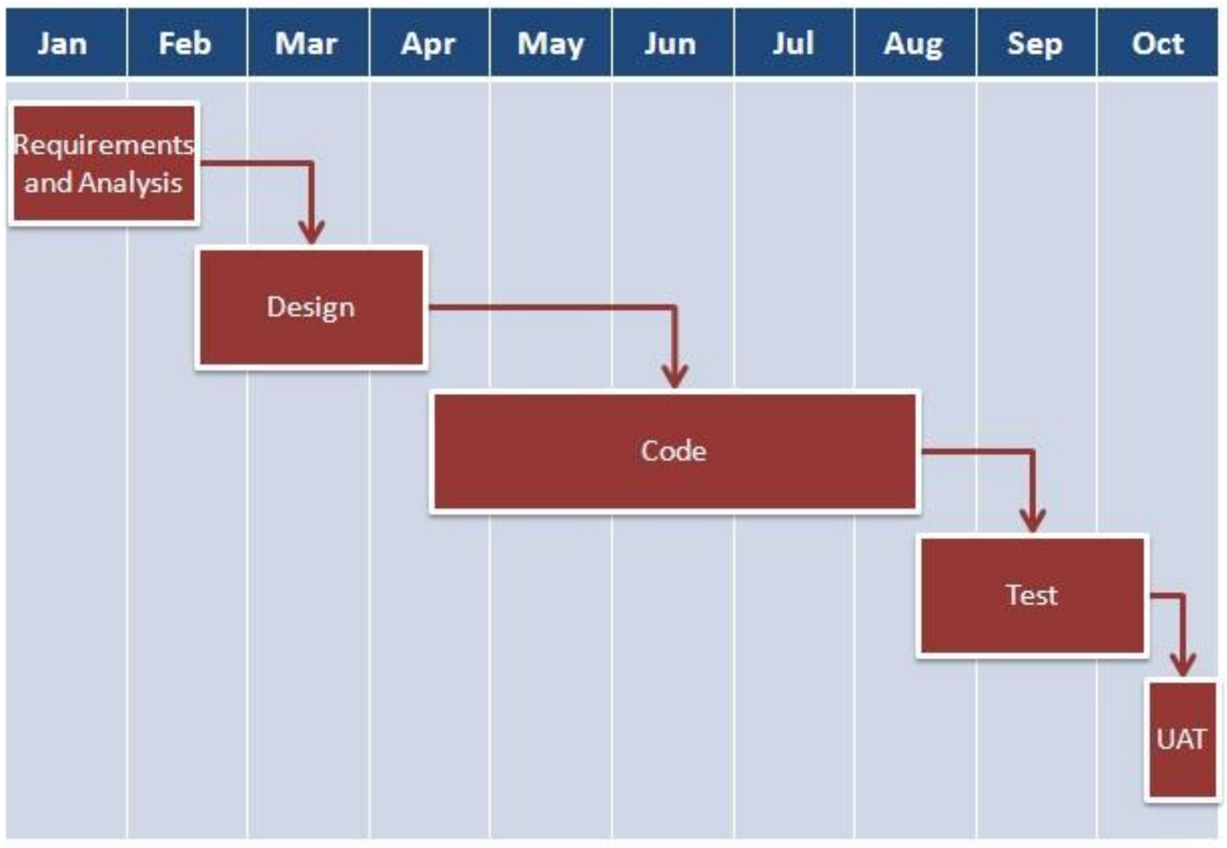
In traditional Waterfall model –

- At a high level, the project teams would spend 15% of their time on gathering requirements and analysis (1.5 months)

Software Development Life Cycle

- 20% of their time on design (2 months)
- 40% on coding (4 months) and unit testing
- 20% on System and Integration testing (2 months).
- At the end of this cycle, the project may also have 2 weeks of User Acceptance testing by marketing teams.
- In this approach, the customer does not get to see the end product until the end of the project, when it becomes too late to make significant changes.

The image below shows how these activities align with the project schedule in traditional software development.



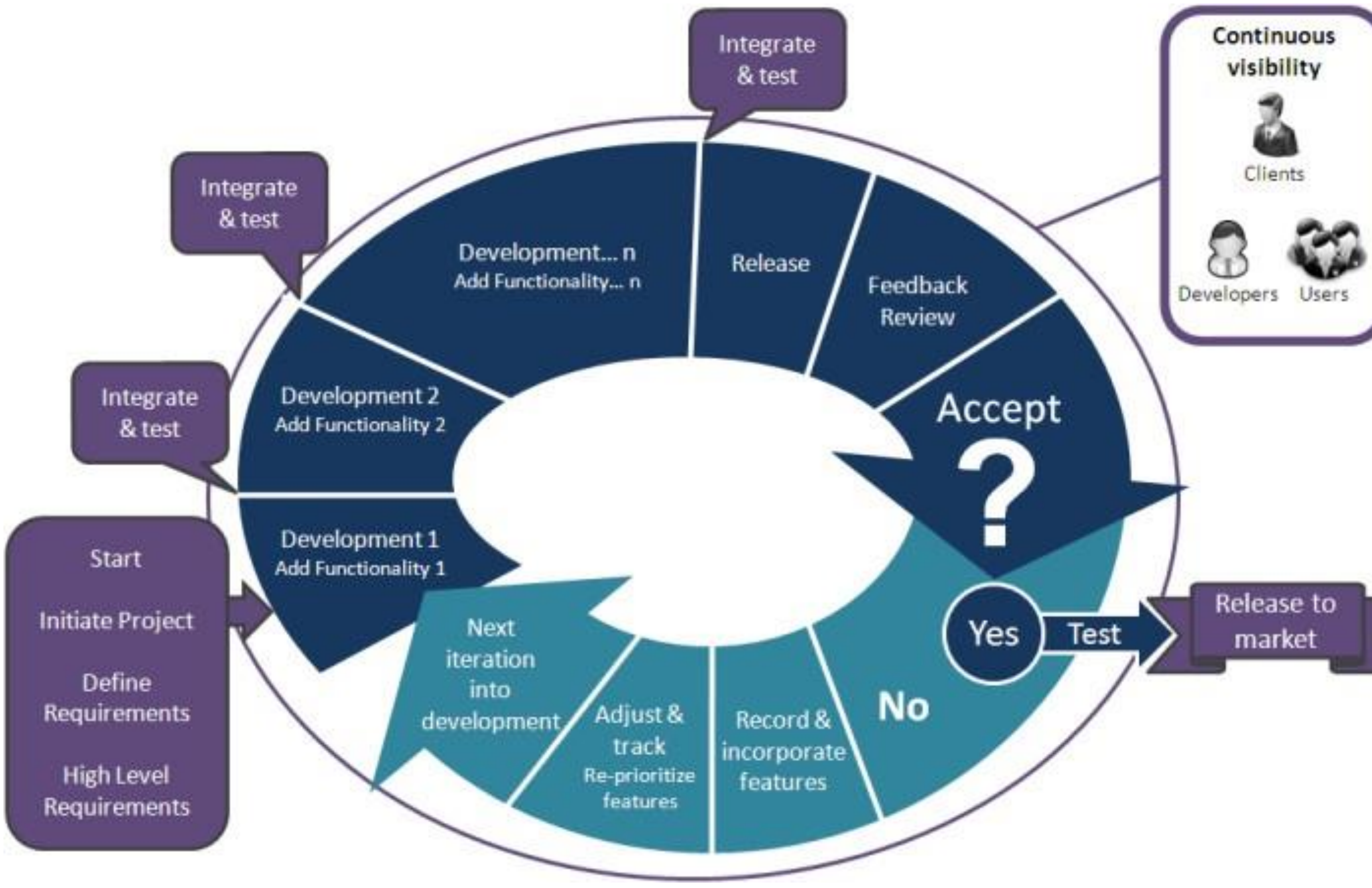
With **Agile development** methodology –

- In the **Agile methodology**, each project is broken up into several 'Iterations'.
- All Iterations should be of the same time duration (between 2 to 8 weeks).
- At the end of each iteration, a working product should be delivered.
- In simple terms, in the Agile approach the project will be broken up into 10 releases (assuming each iteration is set to last 4 weeks).
- Rather than spending 1.5 months on requirements gathering, in Agile software development, the team will decide the basic core features that are required in the product and decide which of these features can be developed in the first iteration.
- Any remaining features that cannot be delivered in the first iteration will be taken up in the next iteration or subsequent iterations, based on priority.
- At the end of the first iterations, the team will deliver a working software with the features that were finalized for that iteration.

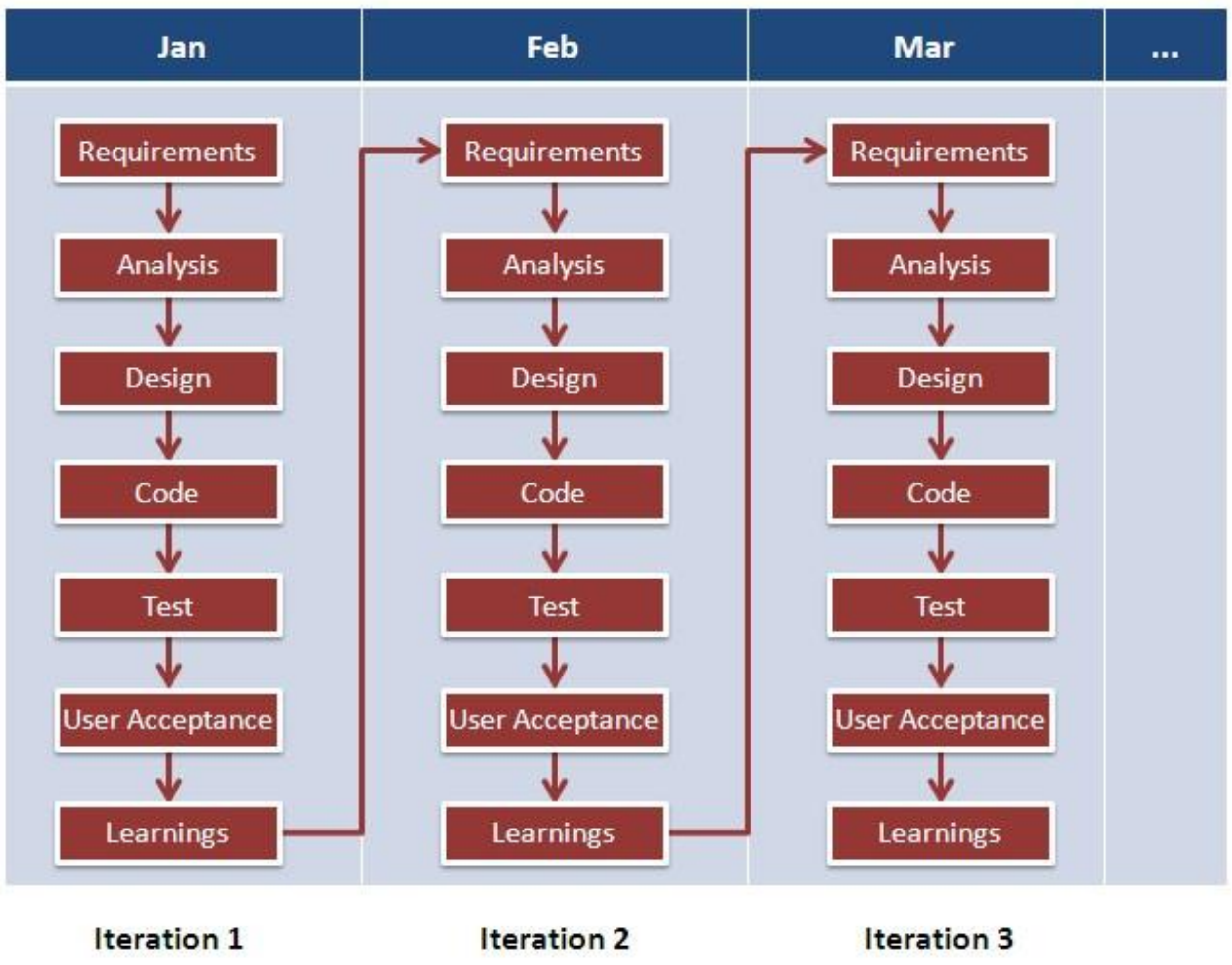
Software Development Life Cycle

- There will be 10 iterations and at the end of each iteration the customer is delivered a working software that is incrementally enhanced and updated with the features that were shortlisted for that iteration.

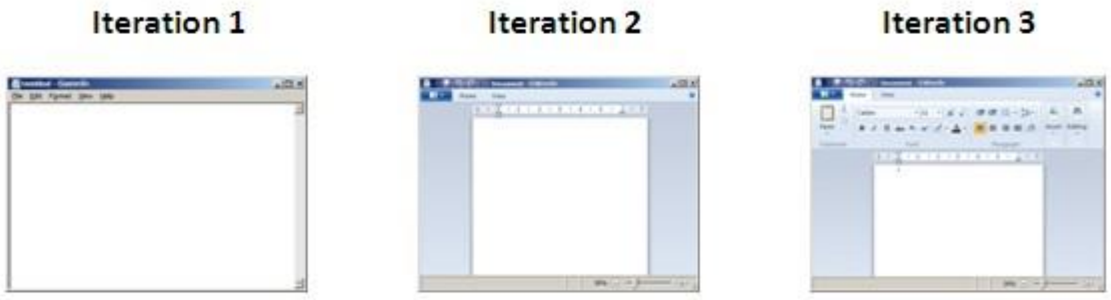
The iteration cycle of an Agile project is shown in the image below.



Software Development Life Cycle



This approach allows the customer to interact and work with functioning software at the end of each iteration and provide feedback on it. This approach allows teams to take up changes more easily and make course corrections if needed. In the Agile approach, software is developed and released incrementally in the iterations. An example of how software may evolve through iterations is shown in the image below.



Agile methodology gives more importance to collaboration within the team, collaboration with the customer, responding to change and delivering working software.

Software Development Life Cycle

- **Who is Scrum Master**

A scrum master might sound like a character in a role-playing game, but it's a serious job that's rooted in leadership. The scrum master is responsible for ensuring a true scrum process over the course of a project. They hold together the scrum framework, facilitating the process for the organization, product owner and scrum team**.** Scrum is a framework that allows teams to work on complex projects and deliver high-value products by approaching problems adaptively. It's a simple, straightforward and easy-to-implement way to handle projects. It can pivot and encourages continuous feedback, which allows a project to more accurately fulfill a customer's needs.

- **Differentiate between Product/Sprint Backlog**

The sprint backlog is like a subset of the product backlog. The sprint backlog comes from the product backlog, but it contains only that item, or those items, that can be completed during each sprint. Think of it as the marching orders for the team as they go off on their short sprint.

The product backlog is compiled of all the things that must be done to complete the whole project. But it's not just a simple list. An effective product backlog breaks down each of the items on the list into a series of steps that helps the development team. There must be a duration, so the team knows when to start the task and how long they have until they must finish it.

- **What is Epic & Story**

Stories, also called "user stories," are short requirements or requests written from the perspective of an end user.

Epics are large bodies of work that can be broken down into a number of smaller tasks (called stories)

- **What is called Velocity in SCRUM**

Velocity is a measure of the amount of work a Team can tackle during a single Sprint and is the key metric in Scrum. Velocity is calculated at the end of the Sprint by totaling the Points for all fully completed user stories

For example, to track Agile velocity, most Scrum teams measure the number of user points in a given sprint. Once this is measured based on a few sprints, the team can then predict how many user points they should plan to complete per sprint. This ultimately reveals how many sprints it will take to complete a project, and helps the team to measure efficiency along the way.

- **Explain the SCRUM ceremonies**

Scrum is executed in what are called sprints, or short iterations of work lasting usually no more than two weeks. A sprint employs four different scrum ceremonies to ensure proper execution: sprint planning, daily scrum, sprint review and sprint retrospective. These scrum ceremonies are outlined below:

- **Sprint Planning:** This is where the team meets and decides what they need to complete in the coming sprint
- **Daily Scrum:** This is a standup meeting, or a very short – 15-minute mini-meeting – for the team to make sure they're all on the same page.
- **Sprint Review:** This is another type of meeting, but one in which the team demos what they shipped in the sprint.
- **Sprint Retrospective:** This is when the team reviews their work, identifying what they did well and what didn't go as planned, so they can make the next sprint better.

- **What is grooming**

Grooming (or refinement) is a meeting of the Scrum team in which the product backlog items are discussed and the next sprint planning is prepared.

Product grooming is critical in product management because it means keeping the backlog up to date and getting backlog items ready for upcoming sprints.

Backlog grooming is often named pre-planning. The product owner and team representatives arrange it in the mid-sprint time. In this case, planning and refinement meetings alternate but happen on the same day each week. That provides an effective rhythm for the entire team.The grooming involves splitting big items into smaller ones, rewriting backlog items to be more expressive, deleting obsolete or no more need items, and so on.

➢ **How Jira board is effective in SCRUM**

A board displays issues from one or more projects, giving you a flexible way of viewing, managing, and reporting on work in progress. There are three types of boards in Jira Software:

- **Next-gen board:** For teams who are new to agile. Get your team up-and-running with this simplified board. The set-up is straight-forward and streamlined, delivering more power progressively as you need it.
- **Scrum board:** For teams that plan their work in sprints. Includes a backlog.
- **Kanban board:** For teams that focus on managing and constraining their work-in-progress. Includes the option of a Kanban backlog.

➢ **Differentiate between SCRUM & Waterfall**

Waterfall	**Scrum**
Schedule driven	Value driven
Plan creates the cost, schedule, estimates	Valued features drive estimates
Development is phase based and sequential	Development is iterative and incremental
Focus is predictive	Focus is adaptive
Demonstrate progress by reporting on activity and stage gateways	Demonstrate progress by delivering valued features every two weeks
Product quality at the end after extensive test/fix activities	Quality is built in with upfront standards
Batches are large (frequently 100%)	Optimises smaller, economically sensible, batch sizes for speed of delivery of valued features
Critical learning applies on one major analyse-design-build-test loop	Leverages multiple concurrent learning loops

Process is tolerant of late learning	Work is organised for fast feedback
Handovers between analyse-design-build-test phases with knowledge stored in documents	Cross-functional team with knowledge of the product invested in the whole team through shared experiences

The main difference in comparing the two methods is:

- Scrum is value driven, where the plan is formed around the question “what is the most valuable item we can deliver today”
- Waterfall produces a plan from which costs, schedule and estimates are created

If we wanted to deliver value to clients and stakeholders, Scrum is the method we would choose.

- **Explain the responsibilities of Product Owner**

The product owner is a role on a product development team responsible for managing the product backlog in order to achieve the desired outcome that a product development team seeks to accomplish. Key activities to accomplish this include:

- Clearly identify and describe product backlog items in order to build a shared understanding of the problem and solution with the product development team
- Make decisions regarding the priority of product backlog items in order to deliver maximum outcome with minimum output
- Determine whether a product backlog item was satisfactorily delivered
- Ensure transparency into the upcoming work of the product development team.

The product owner role was created as part of the Scrum framework in order to address challenges that product development teams had with multiple, conflicting direction, or no direction at all with respect to what to build.

Many infer that a product owner is someone who can spend a considerable amount of time with the product development team providing clarification on product backlog items, and making decisions about which product backlog items to do and regarding the specifics of those particular product backlog items.