

❖ **Agenda**

- ❖ What is Agile?
- ❖ History of Agile
- ❖ Comparison of Waterfall and Agile
- ❖ Methodology
- ❖ Agile Manifesto
- ❖ Overview of Agile Project Management
- ❖ Advantages and Disadvantages of Agile
- ❖ Agile Software Development Approaches
- ❖ Collaborative User Story Creation
- ❖ Continuous Integration, Release, Iteration and Planning

What is Agile?

- Agile development is an umbrella term that describes several agile methodologies to handle IT teams and projects.
- The word 'Agile' is derived from agile manifesto.
- **Agile software development** is a group of software development methods based on iterative and incremental development, where requirements and solutions evolve through collaboration between self-organizing, cross-functional teams.
 - Methods
 - Iterative
 - Incremental

Continue.....

- It promotes adaptive planning, evolutionary development and delivery, a time-boxed iterative approach, and encourages rapid and flexible response to change.
- It is a conceptual framework that promotes foreseen interactions throughout the development cycle.
- The Agile Manifesto[introduced the term in 2001.
(Wiki, 21 Aug 12)

History of Agile

- Some frustrations around seemingly unproductive software development activities, which were shared by like-minded professionals, led to the now-famous Snowbird meeting in Utah in early 2001.
- But that wasn't the first time this particular group of software leaders had met. They had gathered the year before, at the Rogue River Lodge in Oregon in the spring of 2000.
- This group included Kern, Extreme Programming pioneers Kent Beck and Ward Cunningham, Arie van Bennekum, Alistair Cockburn, and twelve others, all well known today in the agile community. Agile, as a practice, was not the ultimate goal; in fact, "agile" had yet to be used in formal conversation before that time.

History of Agile

- At that meeting, the terms "light" and "lightweight" were more common, although none of the participants were particularly satisfied with that description.
- In particular, these thought leaders sought ways to quickly build working software and get it into the hands of end users.
- This fast delivery approach provided a couple of important benefits. First, it enabled users to get some of the business benefits of the new software faster. Second, it enabled the software team to get rapid feedback on the software's scope and direction.

Agile Software Development: Intro

- Characteristics of Agile Software Development
 - Light Weighted methodology
 - Small to medium sized teams
 - vague and/or changing requirements
 - vague and/or changing techniques
 - Simple design
 - Minimal system into production



Characteristics

- Modularity
- Iterative
- Time-bound
- Incremental
- Convergent
- People-oriented
- Collaborative

Comparison of Waterfall and Agile Methodology

- Waterfall Model methodology which is also known as Linear Sequential Life Cycle Model. Waterfall Model followed in the sequential order, and so project development team only moves to next phase of development or testing if the previous step completed successfully.
- Agile methodology is a practice that helps continuous iteration of development and testing in the software development process. In this model, development and testing activities are concurrent, unlike the Waterfall model. This process allows more communication between customers, developers, managers, and testers.

Comparison Of Waterfall and Agile

Agile Methodology	Waterfall Methodology
It separates the project development lifecycle into sprints.	Software development process is divided into distinct phases.
It follows an incremental approach	Waterfall methodology is a sequential design process.
Agile methodology is known for its flexibility.	Waterfall is a structured software development methodology so most times it can be quite rigid.
Agile can be considered as a collection of many different projects.	Software development will be completed as one single project.
Agile is quite a flexible method which allows changes to be made in the project development requirements even if the initial planning has been completed.	There is no scope of changing the requirements once the project development starts.

Comparison Of Waterfall and Agile

Agile Methodology	Waterfall Methodology
Agile methodology, follow an iterative development approach because of this planning, development, prototyping and other software development phases may appear more than once.	All the project development phases like designing, development, testing, etc. are completed once in the Waterfall model.
Test plan is reviewed after each sprint	The test plan is rarely discussed during the test phase.
Agile development is a process in which the requirements are expected to change and evolve.	The method is ideal for projects which have definite requirements and changes not at all expected.
In Agile methodology, testing is performed concurrently with software development.	In this methodology, the "Testing" phase comes after the "Build" phase

Comparison of Waterfall and Agile

Agile Methodology	Waterfall Methodology
Agile introduces a product mindset where the software product satisfies needs of its end customers and changes itself as per the customer's demands.	This model shows a project mindset and places its focus completely on accomplishing the project.
Agile methodology works exceptionally well with Time & Materials or non-fixed funding. It may increase stress in fixed-price scenarios.	Reduces risk in the firm fixed price contracts by getting risk agreement at the beginning of the process.
Prefers small but dedicated teams with a high degree of coordination and synchronization.	Team coordination/synchronization is very limited.

Comparison of Waterfall and Agile

Agile Methodology	Waterfall Methodology
Products owner with team prepares requirements just about every day during a project.	Business analyst prepares requirements before the beginning of the project.
Test team can take part in the requirements change without problems.	It is difficult for the test to initiate any change in requirements.
Description of project details can be altered anytime during the SDLC process.	Detail description needs to implement waterfall software development approach.
The Agile Team members are interchangeable, as a result, they work faster. There is also no need for project managers because the projects are managed by the entire team	In the waterfall method, the process is always straightforward so, project manager plays an essential role during every stage of SDLC

Agile Manifesto

www.agilemanifesto.org

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

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

That is, while there is value in the items on the right, we value the items on the left more.

Principles behind the Agile Manifesto

1. Our highest priority is to **satisfy the customer** through early and **continuous delivery of valuable software**.
2. **Welcome changing requirements**, even late in development. Agile processes **harness change** for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must **work together** daily throughout the project.
5. Build projects around **motivated individuals**. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.

Principles behind the Agile Manifesto

7. **Working software** is the primary **measure** of progress.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to **technical excellence** and **good design** enhances agility.
- 10 **Simplicity**--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
- 12 At regular intervals, the **team reflects** on how to become more effective, then tunes and adjusts its behavior accordingly.

Overview of Agile Project Management

- Agile methodology is a type of project management process, mainly used for software development, where demands and solutions evolve through the collaborative effort of self-organizing and cross-functional teams and their customers.
- Agile project management is a methodology that is commonly used to deliver complex projects due to its adaptiveness. It emphasizes collaboration, flexibility, continuous improvement, and high quality results. It aims to be clear and measurable by using six main “deliverables” to track progress and create the product.

The deliverables

- **Product vision statement:** A summary that articulates the goals for the product.
- **Product roadmap:** The high-level view of the requirements needed to achieve the product vision.
- **Product backlog:** Ordered by priority, this is the full list of what is needed to be done to complete your project.
- **Release plan:** A timetable for the release of a working product.
- **Sprint backlog:** The user stories (requirements), goals, and tasks linked to the current sprint.
- **Increment:** The working product functionality that is presented to the stakeholders at the end of the sprint, and could potentially be given to the customer.

Overview of Agile Project Management

- There are various frameworks within Agile project management that can be used to develop and deliver a product or service. While they each have their own set of characteristics and terminology, they share common principles and practices.
- Two of the most popular ones that support the Agile development life cycle are Scrum and Kanban.

Advantages of Agile

- In Agile methodology the delivery of software is unremitting.
- The customers are satisfied because after every Sprint working feature of the software is delivered to them.
- If the customers has any feedback or any change in the feature then it can be accommodated in the current release of the product.
- In Agile methodology the daily interactions are required between the business people and the developers.
- In this methodology attention is paid to the good design of the product.
- Changes in the requirements are accepted even in the later stages of the development.

Disadvantages of Agile

- In Agile methodology the documentation is less.
- Sometimes in Agile methodology the requirement is not very clear hence it's difficult to predict the expected result.
- In few of the projects at the starting of the software development life cycle it's difficult to estimate the actual effort required.
- The projects following the Agile methodology may have to face some unknown risks which can affect the development of the project.

Agile Software Development Framework

- Several Agile frameworks are existing in the commercial market which are being widely used by the organizations.
- All these agile framework embrace Agile manifesto, Agile principles and values.
- An **agile framework** can be **defined** as a specific software-development approach based on the **agile** philosophy articulated in the **Agile** Manifesto. You can refer to any of these **frameworks** as methodologies or even processes.

Agile Software Development Framework

- The following section describes the agile software development approaches in detail.
 - Scrum
 - Kanban
 - Extreme Programming(XP)
 - Lean
 - Crystal
 - Dynamic System Driven Methodology (DSDM)
 - Feature Driven Development

Collaborative User Stories

- User stories are part of an agile approach that helps shift the focus from writing about requirements to talking about them.
- All agile user stories include a written sentence or two and, more importantly, a series of conversations about the desired functionality.
- **User stories** are short, simple descriptions of a feature told from the perspective of the person who desires the new capability, usually a user or customer of the system. They typically follow a simple template:
- As a < type of user >, I want < some goal > so that < some reason >.

Collaborative User Stories

- User stories are often written on index cards or sticky notes, stored in a shoe box, and arranged on walls or tables to facilitate planning and discussion.
- As such, they strongly shift the focus from writing about features to discussing them. In fact, these discussions are more important than whatever text is written.

Collaborative User Story

- Traditional requirements specifications are usually created by business analyst and validated by business stakeholders.
- This approach can result in low quality specifications as users lack insight into their true needs.
- Absence of a global vision for the system, redundant or contradictory features, and other miscommunication can further degrade the specifications.
- Poor specifications are often a major reason for project failure.

Collaborative User Stories

- In Agile development, user stories are written to capture requirements from the perspectives of developers, testers, and business representatives.
- In sequential development, this shared vision of a feature is accomplished through formal reviews after requirements are written.
- In Agile development, this shared vision is accomplished through frequent informal reviews while the requirements are being written.
- **Example:**
- As an **online bank customer**, I want to **access my bank statement for the last 5 years**

Component elements of User story

- Ron Jefferies proposed the 3C concept to describe the component elements of a user story. The three elements are card, conversation and confirmation:
- **Card:** The card is the physical media which describes a user story. It identifies the requirement, its criticality and the acceptance criteria.
- It also includes expected development and test duration. This card is used in the product backlog so the description has to be accurate.

Component elements of User Story

- **Conversation:** The requirements are communicated and understood, in a number of conversations conducted between the customers, developers and testers.
- The conversation explains how the software will be used. The conversation can be documented or verbal.
- The tester perspective adds much to this conversation by asking questions, identifying ambiguity, missing elements such as non-functional aspects and suggesting ways to test the story.
- Conversation begins during the release-planning phase and continues when the story is scheduled for inclusion in iteration.

Component elements of User Story

- **Confirmation:** The acceptance criteria, discussed in the conversation, are used to confirm that the story is done.
- These acceptance criteria may span more than one user stories. Positive and negative tests should be used to cover the criteria.
- To confirm that the story is done, the defined acceptance criteria should be tested and satisfied.

Collaborative User Story

- In nutshell, the **user stories** must address both **functional** and **non-functional** characteristics.
- Each story includes **acceptance criteria** for these characteristics.
- These criteria should be defined in **collaboration** between **business representatives**, **developers**, and **testers**.
- They provide developers and testers with an extended vision of the feature that business representatives will validate.

Collaborative User Story

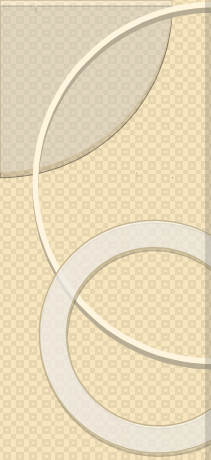
- An Agile team considers a task finished when **a set of acceptance criteria** have been satisfied.
- Typically, the tester's unique perspective will improve the user story by identifying **missing details or non-functional requirements**.
- A tester can contribute by asking business representatives **open-ended questions** about the user story, proposing ways to test the user story, and confirming the acceptance criteria.

Collaborative User Story

- The collaborative authorship of the user story can use techniques such as brainstorming and mind mapping. The tester may use the INVEST technique:
- **I**ndependent: discrete or self-contained, it does not rely on another user story
- **N**egotiable: always subject to change
- **V**aluable: of value to the business or customer
- **E**stimable: can be estimated
- **S**mall: small enough to be planned as part of an iteration
- **T**estable: has acceptance criteria that can be used to create tests to ensure the software is correct

Collaborative User Story

- Agile teams vary in terms of how they document user stories.
- Regardless of the approach taken to document user stories, documentation should be concise, sufficient, and necessary.



Continuous Integration, Delivery, Development, Iteration and Planning

Developers practicing continuous integration merge their changes back to the main branch as often as possible. The developer's changes are validated by creating a build and running automated tests against the build.

By doing so, you avoid the integration hell that usually happens when people wait for release day to merge their changes into the release branch.

Continuous integration puts a great emphasis on testing automation to check that the application is not broken whenever new commits are integrated into the main branch.



Continuous Integration, Delivery, Development, Iteration and Planning

Continuous delivery is an approach where teams release quality products frequently and predictably from source code repository to production in an automated fashion.

Continuous delivery is an extension of continuous integration to make sure that you can release new changes to your customers quickly in a sustainable way. This means that on top of having automated your testing, you also have automated your release process and you can deploy your application at any point of time by clicking on a button.

In theory, with continuous delivery, you can decide to release daily, weekly, fortnightly, or whatever suits your business requirements. However, if you truly want to get the benefits of continuous delivery, you should deploy to production as early as possible to make sure that you release small batches that are easy to troubleshoot in case of a problem.

Continuous Deployment

Continuous deployment solves this problem by automatically shipping every change pushed to the main repository to production. It's a radical approach – very different from spending days preparing and controlling each release – but there are several benefits to continuous deployment:

- Releases become smaller and easier to understand.
- No one is required to drop their work to make a deployment because everything is automated.
- The feedback loop with your customers is faster: new features and improvements go straight to production when they're ready.
- If you want to implement a continuous deployment pipeline, the best place to start with continuous delivery. These two practices are similar, but differ in their approaches to production deploys.



Continuous Deployment

In continuous delivery, every change pushed to the main repository is ready to be shipped, but the production release process still requires human approval.

In continuous deployment, the release to production is done automatically for every change that passes the test suite.



Continuous Integration, Delivery, Development, Iteration and Planning

Continuous Iteration: an iteration is a single development cycle, usually measured as one week or two weeks. An iteration may also be defined as the elapsed time between iteration planning sessions.

While the adjective iterative can be used to describe any repetitive process, it is often applied to any heuristic planning and development process where a desired outcome, like a software application, is created in small sections.

These sections are iterations. Each iteration is reviewed and critiqued by the software team and potential end-users; insights gained from the critique of an iteration are used to determine the next step in development.

Continuous Integration, Delivery, Development, Iteration and Planning

Continuous Planning: There are 3 levels of planning in Agile. They are Release Planning, Iteration Planning and Daily Planning. These planning meetings help the **Scrum Master, Product Owner** and the rest of the team in understanding how the product will be delivered, the complexity involved and their day to day responsibility in the delivery of the product, among other things.