SOFTWARE ENGINEERING

UNIT – 1

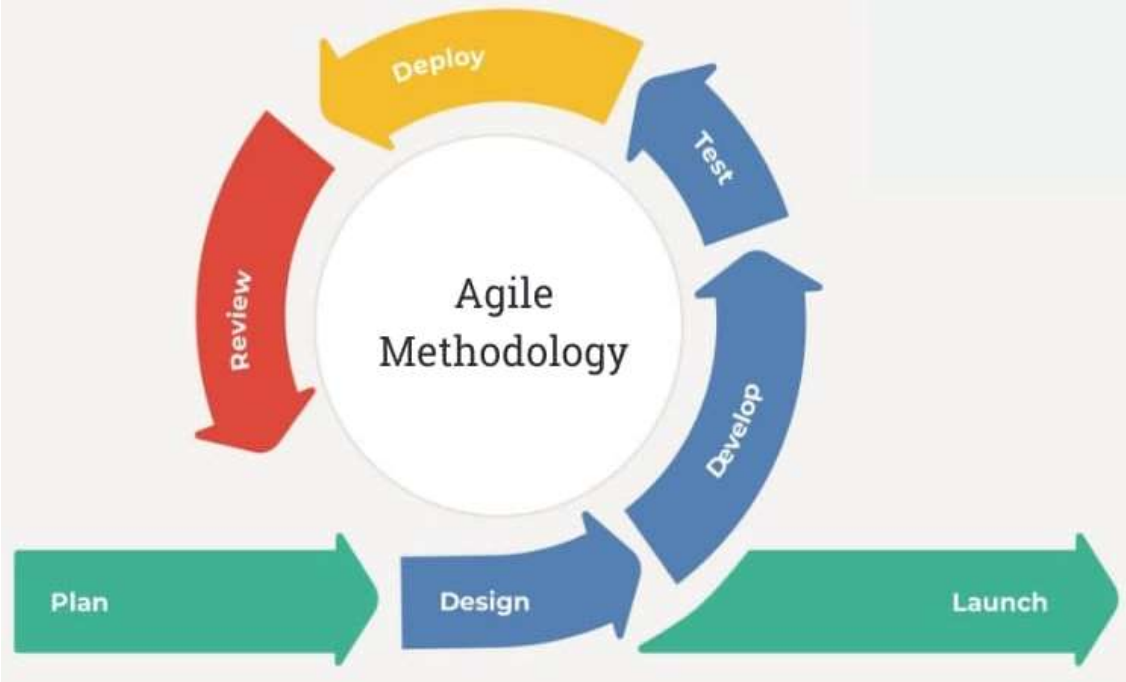
TOPIC – 7

INTRODUCTION TO AGILE

1. What is Agile?

Agile is a flexible approach to developing software. Unlike traditional methods where everything is planned in advance, Agile allows teams to adapt and make changes as they go.

Example: If you're building a mobile app, instead of planning everything from the start, you would release a basic version. Then, based on user feedback, you keep improving the app over time.



2. Agile Manifesto

The **Agile Manifesto** is a set of values and principles created in 2001 that guides how software should be developed. The Agile Manifesto consists of **64 words** that define the key principles of Agile software development.

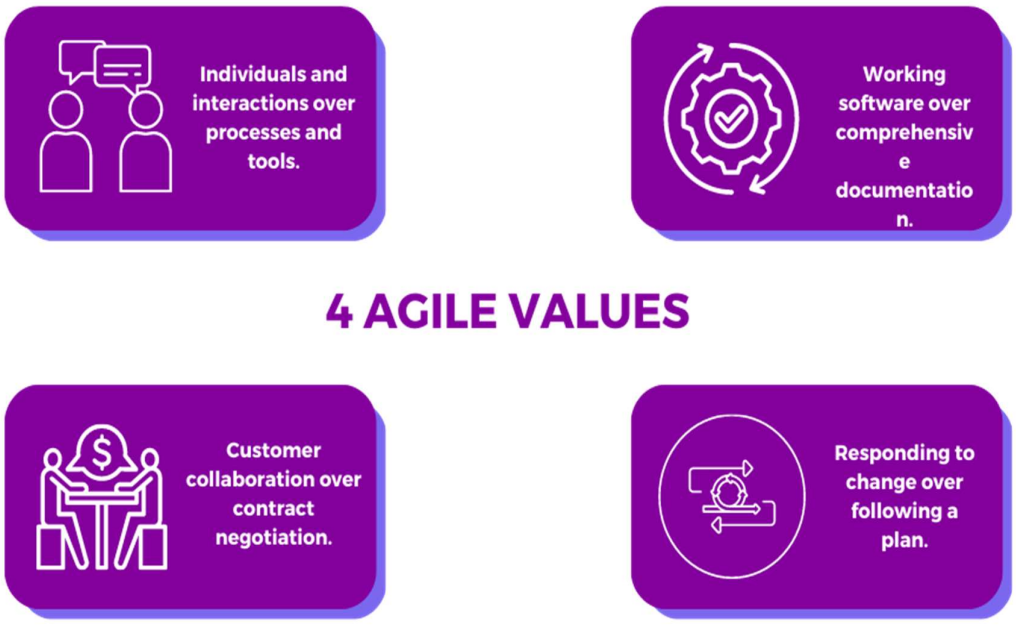
The **64 words** in the Agile Manifesto are the foundation of Agile methodology. Here they are:

"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."

Four Key Values:



1. **Individuals and Interactions over Processes and Tools**
   - Agile prioritizes direct communication among team members rather than rigidly following processes.
   - **Example:** A developer can quickly ask another team member for clarification instead of waiting for formal documentation.

2. **Working Software over Comprehensive Documentation**
   - Focus on getting the software working rather than spending months writing detailed plans.
   - **Example:** Instead of writing a long report about how a feature should work, build the feature and let the users test it.
3. **Customer Collaboration over Contract Negotiation**
   - Work closely with the customer to understand their changing needs rather than sticking strictly to the initial contract.
   - **Example:** If the customer decides they need a new feature halfway through the project, you can adjust the work accordingly.
4. **Responding to Change over Following a Plan**
   - Be open to making changes, even if it means altering the original plan.
   - **Example:** If a new technology comes up that can improve the project, Agile allows you to adopt it, even if it wasn't planned from the start.

12 Agile Principles

These principles provide more specific guidance on how to implement the Agile values:

1. **Customer satisfaction** is the highest priority, achieved through early and continuous delivery of software.
   - **Example:** Release new features or updates every few weeks, so the customer always has something new to try.
2. **Welcome changing requirements**, even late in development.
   - **Example:** If the customer wants a change in the middle of development, it's welcomed rather than avoided.
3. **Deliver working software frequently**.
   - **Example:** Instead of waiting months, you release working software every two weeks (a cycle known as a "sprint").
4. **Daily collaboration** between business people and developers.
   - **Example:** The product owner (representing the customer) talks to the developers every day to make sure everyone is on the same page.
5. **Build projects around motivated individuals** and give them the freedom to make decisions.
   - **Example:** The development team is trusted to make technical decisions without waiting for approval from higher management.
6. **Face-to-face communication** is the best way to convey information.
   - **Example:** Instead of sending emails or creating documents, teams should hold short daily meetings to discuss progress and issues.
7. **Working software** is the primary measure of progress.
   - **Example:** The project is judged based on how much of the software is actually usable, rather than how much documentation or planning has been done.
8. **Sustainable development** means the team should be able to maintain a constant pace without burnout.
   - **Example:** Teams avoid working overtime constantly to ensure steady progress over time.
9. **Continuous attention to technical excellence** and good design enhances agility.
   - **Example:** Developers regularly refactor the code to keep it clean and maintainable, which makes future changes easier.
10. **Simplicity**, or the art of maximizing the amount of work not done, is essential.
    - **Example:** Focus only on building features that provide value, rather than adding unnecessary details.
11. **Self-organizing teams** produce the best designs and architectures.

- **Example:** Instead of following a strict hierarchy, the team decides the best way to achieve the project goals together.

12. **Regular reflection** on how to become more effective.
    - **Example:** After each sprint, the team holds a "retrospective" to discuss what went well and what could be improved.

3. Advantages of Agile

1. **Increased Customer Satisfaction**
    - Continuous feedback and updates keep the customer happy.
    - **Example:** The customer feels involved as they can see regular progress and suggest changes.
2. **Faster Delivery of Software**
    - Short development cycles mean features are delivered faster.
    - **Example:** Customers receive working software every few weeks, rather than waiting for months.
3. **Flexibility to Change**
    - Agile welcomes changes, even in later stages.
    - **Example:** If a new market trend or technology emerges, Agile allows the team to adjust their plans.
4. **Improved Product Quality**
    - Constant testing and feedback result in a more refined product.
    - **Example:** The product is tested at the end of each sprint, catching issues early.
5. **Enhanced Team Collaboration**
    - Agile promotes regular communication and teamwork.
    - **Example:** Daily meetings help everyone stay on the same page, improving overall team dynamics.
6. **Predictable Costs and Schedule**
    - Agile's regular sprints help teams manage time and budget more effectively.
    - **Example:** With fixed timeframes, it's easier to predict the costs and when a feature will be delivered.
7. **Early Risk Identification and Mitigation**
    - Problems are identified early due to frequent testing.

- **Example:** By releasing small updates, teams can spot potential issues early and adjust the project accordingly.

4. Disadvantages of Agile

1. **Not Suitable for Highly Complex Projects**

- Agile may struggle when strict planning and documentation are required.

2. **Requires Strong Leadership**

- A skilled leader is necessary to ensure the project stays on track.

3. **Tight Deadlines Can Be Challenging**

- Meeting frequent deadlines might result in rushed decisions or sacrifices in quality.

4. **Lack of Documentation**

- Since Agile relies on communication over documentation, it can be harder to onboard new team members or refer back to old decisions.

5. Agile Frameworks

Agile isn't a specific method but rather a collection of frameworks. Some popular ones include:

1. **Scrum**: A framework that uses iterative development with time-boxed iterations called sprints, typically lasting 2-4 weeks. It focuses on delivering small, incremental improvements.
2. **Kanban**: A visual management method that uses a board with columns to represent different stages of work. It emphasizes continuous delivery and improvement by managing work in progress.
3. **Extreme Programming (XP)**: A framework focused on technical excellence and customer satisfaction. It emphasizes practices like continuous integration, test-driven development, and pair programming.
4. **Lean Software Development**: Inspired by lean manufacturing principles, this framework focuses on eliminating waste, improving flow, and maximizing value delivered to the customer.

5. **Feature-Driven Development (FDD)**: A model-driven, short-iteration process that focuses on designing and building features. It emphasizes creating a feature list and then delivering those features incrementally.
6. **Crystal**: A family of methodologies tailored to different team sizes and project complexities. It emphasizes flexibility and the use of frequent delivery and reflective improvement.
7. **Dynamic Systems Development Method (DSDM)**: A framework that provides a structured approach to project management, focusing on delivering business value through iterative development and active user involvement.
8. **Agile Unified Process (AUP)**: A simplified version of the Rational Unified Process (RUP), AUP integrates Agile practices into the traditional Unified Process framework, focusing on iterative development and continuous improvement.
9. **Scaled Agile Framework (SAFe)**: A framework designed to scale Agile practices to larger organizations or projects. It includes principles and practices for aligning teams, programs, and portfolios.
10. **Large Scale Scrum (LeSS)**: An extension of Scrum for scaling Agile across multiple teams, maintaining the simplicity and principles of Scrum while addressing coordination and integration across teams.

    Each framework has its own unique approach and practices, so the choice often depends on the specific needs and context of the organization or project.