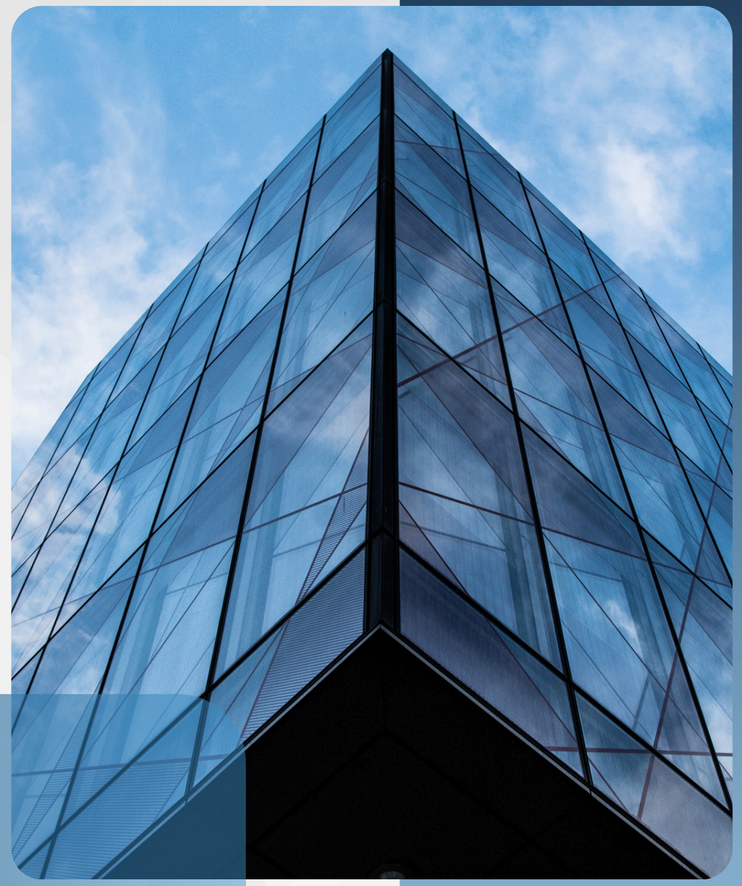
Agile Methodology

Presented by Nandhiha V



Agile Software Development — At a Glance

Agile is an iterative and incremental approach to software development that emphasizes collaboration, adaptability, and customer satisfaction.

Core Values (Agile Manifesto):

- Individuals & Interactions over processes & tools
- Working Software over comprehensive documentation
- Customer Collaboration over contract negotiation
- Responding to Change over following a plan

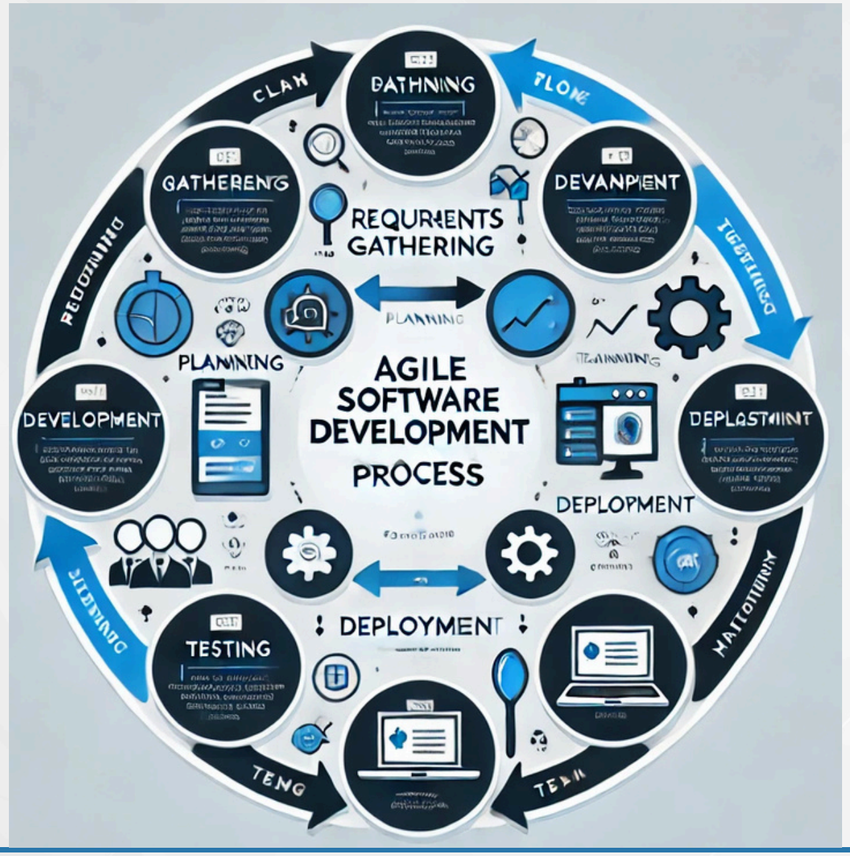
Why Agile is Used?

- Delivers tangible value early
- Focuses on value-added tasks
- Encourages team empowerment & positivity
- Enables quick response to change
- Promotes transparency through regular demos
- Encourages cross-functional, self-organizing teams

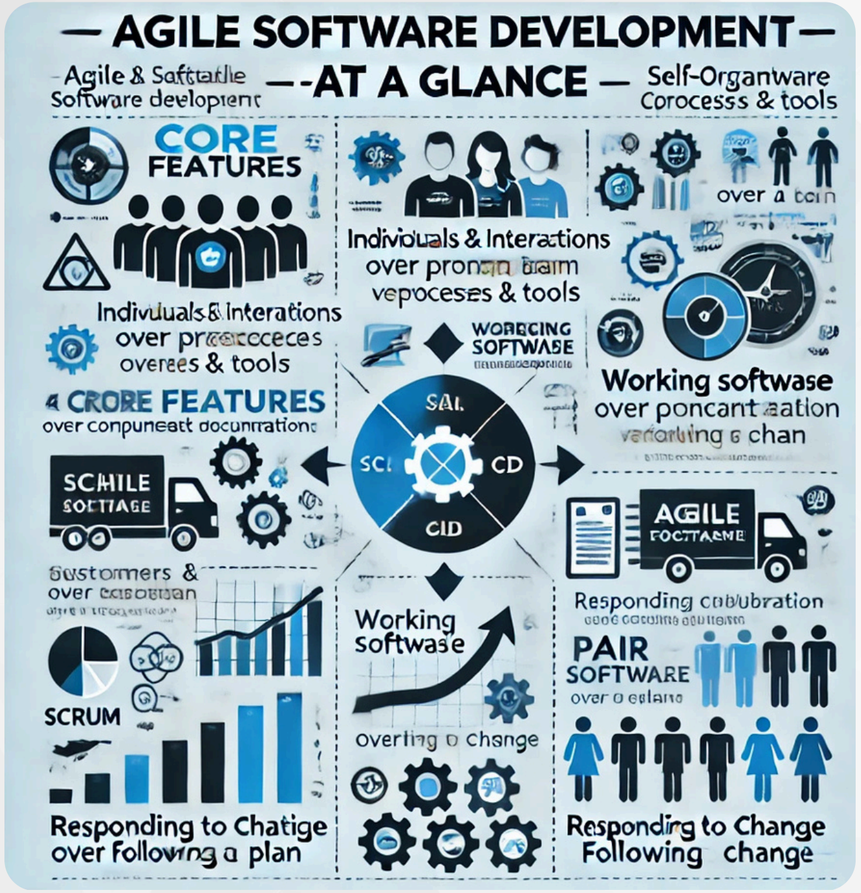
12 Principles of Agile Methodology

- Early and continuous delivery for customer satisfaction
- Embrace changing requirements
- Deliver working software frequently
- Business & developers must work together
- Build projects around motivated individuals
- Face-to-face communication is best
- Working software is the main progress measure
- Maintain a sustainable development pace
- Focus on technical excellence
- Simplicity is essential
- Self-organizing teams create the best results
- Reflect and adjust regularly for effectiveness

Agile Software Development Process



- Requirements Gathering
- Planning
- Development
- Testing
- Deployment
- Maintenance



Agile Development Cycle

- **Concept** – Identify business opportunity
- **Inception** – Team setup, funding, high-level planning
- **Iteration/Construction** – Build software in cycles (sprints)
- **Release** – Deliver working version
- **Production** – Use and support software
- **Retirement** – System shutdown after replacement

Advantages of Agile

- Faster delivery & better customer trust
- Quick response to changes
- Continuous feedback for improvement
- Emphasis on people over process
- Better collaboration & morale
- Improved product quality
- Enhanced customer satisfaction

Disadvantages of Agile

- Difficult effort estimation for large projects
- Less documentation
- Heavily dependent on customer input
- Communication challenges in large teams
- Not ideal for inexperienced developers
- Risk of scope creep
- Team burnout possible
- Lack of structure in some cases

THANK YOU!