1| Agile Methods

↳ Reduce overheads and deliver high quality, flexible Software.

1. Focus on coding, not excessive Planning / Documents
2. Follow an iterative approach – Small, Quick Cycles
3. Deliver working Software quickly and then evolve it with feedback
4. Respond fast to changing Requirements

2| The agile Manifesto

↳ Documentation are important but People, Collaboration, adoptability matter more.

- Individuals and interactions over → process and Tools
- Working Software over → Comprehensive Documentation
- Customer Collaboration over → Contract Negotiation
- Responding to a change over → Following a plan.

3| Principles of Agile Methods

1. Customer involvement: Customer works closely with developers to give feedback and Set Priorities.
2. Incremental delivery: Software is delivered in Small, useful parts
3. People Not process: Team members decide how they work best Trusting their Skills
4. Embrace change: Accept that requirements will change and designing to handle that
5. Maintain Simplicity: Keep Software and process Simple, remo unnecessary Complexity

4| When to use agile

Suitable for

1. Small – Medium Sized Projects
2. Teams that Can work closely with Customers
3. Projects with changing requirements

Not ideal for

1. large – high regulated Sys.
2. projects requiring heavy documentation or high approval Cycles.

5| problems with agile

1. Customers may lose interest over time
2. Not all Customers enjoy the Constant Collaboration style
3. Prioritizing changes Can be difficult with many Stakeholders
4. Maintaining Simplicity takes effort
5. Contracts and legal requirements Can Conflict with the Flexible agile Style.

6| Agile Vs. Plan driven development

Feature	Plan-Driven	Agile
Approach	Sequential Stages	Iterative, overlapping Stages
Planning	Done in advance	Continuous, Flexible
changes	Hard to accomodate	easy to adapt
Documentation	Heavy	Light
Customer involvement	limited	Continuous
Best for	Stable, Regulated Systems	Dynamic, changing Project

7| Choosing between agile and plan driven

depends on:

1. Size of System and team
2. Need for detailed design
3. Skill level of the team
4. lifetime of the System
   - long lifetime → More documentation
5. Team Structure
   - Distributed teams need more Coordination
6. Cultural environment
   - Some Companies prefer Structured planning
7. Regulatory Constraints
   - Government, Medical → Need documentation.

8| Extreme Programming

↳ The most popular agile method.

- Focus on extreme levels of iteration and testing
- New versions built multiple times per day
- Deliver a new working versions to the Customer every 2 weeks
- All tests must pass for every build

– XP Principles:

1. Incremental development: Frequent Small releases
2. Customer involvement: Customer works daily with the team
3. People over process: Team Work, Pair programming
4. Support change: Regular updates
5. Keep it Simple: Continuous refractoring (improving Code)

9| XP Practices

a. Development Practices:

1. Incremental planning: Requirements written on Story Cards, Prioritized by Customer
2. Small Releases: Deliver basic working versions
3. Simple design: only design what's needed
4. Test first development: Write tests before Coding
5. Refactoring: Continuously improve and Simplify the Code

b. Team Practices:

6. Pair Programming: 2 programmers work together on one machine
7. Collective ownership: Anyone Can change any part of the Code
8. Continuous integration: Every Small change is tested and merged
9. Sustainable Pace: No long overtime
10. On Site Customer: Customer representative works directly with the team

10| XP and change

↳ Doesn't design for future changes, instead it keeps improving Code regularly to make change easier

– Keeping Software ① clean ② understandable ③ flexible

11| Refactoring Examples

1. Removing duplicate Code
2. Renaming Confusing functions
3. Reorganizing class hierarchy
4. Replacing repeated Code with reusable functions.

12| Testing in XP

1. Test first improvement: Write tests before writing code
2. user involvement: Customer helps create acceptance tests
3. Automated Testing: Use frameworks to run tests automatically
   ↳ JUnit. after every change

- Advantages: Immediate error detection and Continuous validation
- challenges: Some Parts are hard to test automatically
  ↳ user interface