# AGILE METHODOLOGY

ADAPT • COLLABORATE • DELIVER • IMPROVE

- CUSTOMER COLLABORATION
- ADAPTIVE PLANNING
- CONTINUOUS IMPROVEMENT
- DELIVER VALUE FASTER







PREPARED BY

## V. NIRMALA

ASST PROFESSOR

AIML DEPARTMENT

TEAMWORK QUALITY FLEXIBILITY DELIVERY

INNOVATE • LEARN • COLLABORATE • SUCCEED

| | | |
|---|---|---|
| Title of the Course | : | AGILE METHODOLOGIES |
| Category | : | PE |
| Year | : | III |
| Semester | : | I |
| Course Code | : | 24ACAI5AT |
| Branch/es | : | CSE(AI) |

| Lecture Hours | Tutorial Hours | Practice Hours | Credits | Total Hours |
|---|---|---|---|---|
| 3 | 0 | 0 | 3 | 50 |

**Course Objectives:**

- Knowledge on concepts of Agile development, releasing, planning and developing.

**Course Outcomes:**

At the end of the course, the student will be able to

1. Explain agile principles and Extreme Programming concepts, roles, life cycle, and adoption practices.
2. Demonstrate effective collaboration through customer involvement, team communication, and standard agile practices.
3. Execute reliable software releases using version control, continuous integration, and quality-focused practices.
4. Organize agile planning activities including releases, iterations, risk management, and estimation.
5. Construct software using XP development practices such as TDD, refactoring, and incremental design.

**Unit 1 Introduction Extreme Programming (XP) - Agile Development** **12**

Why Agile - Understanding Success, Beyond Deadlines, Importance of Organizational Success, Introduction to Agility How to Be Agile - Agile methods, Don't make your own method, Road to mastery Understanding XP (Extreme Programming) - XP life cycle, XP team, XP Concepts Adopting XP - Knowing whether XP is suitable, Implementing XP, assessing Agility Practicing XP - Thinking - Pair Programming, Energized work, Informative Workspace, Root cause Analysis, Retrospectives

**Unit 2 Collaborating** 9

Trust, Sit together, Real customer involvement, Ubiquitous language, meetings, coding standards, Iteration demo, Reporting

**Unit 3 Releasing** 9

Bugfree Release, Version Control, fast build, continuous integration, Collective ownership, Documentation

**Unit 4 Planning** **9**

Version, Release Plan, Risk Management, Iteration Planning, Slack, Stories, Estimating.

**Unit 5 Developing** 11

Incremental requirements, Customer tests, Test driven development, Refactoring, Incremental design and architecture, spike solutions, Performance optimization, Exploratory testing.

**Textbook:**


1. The art of Agile Development, James Shore and Shane Warden, 11th Indian Reprint, O'Reilly,2018.

**Reference Books:**


1. Learning Agile, Andrew Stellman and Jennifer Greene, O'Reilly, 4th Indian Reprint, 2018.
2. Practices of an Agile Developer, Venkat Subramaniam and Andy Hunt, SPD, 5th Indian Reprint, 2015.
3. Agile Project Management - Jim Highsmith, Pearson Low price Edition 2004.

## UNIT 1: Introduction to Extreme Programming (XP) – Agile Development

### 1. What is Agile?

Agile is a **software development methodology** that develops software in **small parts (iterations)** instead of completing the entire project at once. It focuses on customer satisfaction, teamwork, flexibility, and continuous improvement.

**Explanation**

Earlier, software was developed using traditional methods like the Waterfall model, where all requirements had to be fixed before development. If customer requirements changed later, it was difficult and expensive to make changes.

Agile solves this problem by allowing developers to:

- Develop software in small releases.
- Get customer feedback after every release.
- Make changes quickly.
- Deliver high-quality software faster.

**Example**

A food delivery app first releases only login and restaurant search. Later, payment, live tracking, and ratings are added based on customer feedback.

**Advantages**

- Fast delivery
- Better quality
- Customer satisfaction
- Easy to manage changes
- Reduced project risk

### 2. Why Agile?

Agile is used because today's business requirements change frequently, and software must adapt quickly.

Customers often change their requirements during development. Agile accepts these changes instead of rejecting them.

**Reasons for using Agile**

- Delivers software quickly.
- Welcomes changing requirements.
- Improves communication.
- Reduces development cost.

- Produces better-quality software.

**Real-time Example**

WhatsApp regularly adds new features like Communities, Channels, and Message Edit without rebuilding the entire application.

**3. Understanding Success**

**Definition**

In Agile, success is measured by customer satisfaction rather than simply completing the project on time.

A successful Agile project:

- Meets customer requirements.
- Delivers quality software.
- Is completed within budget.
- Allows future improvements.

**Example**

Amazon continuously improves its website based on customer reviews and shopping behavior.

**4. Beyond Deadlines**

Traditional development mainly focuses on finishing before the deadline.

Agile focuses on:

- Customer value
- Product quality
- Team collaboration
- Continuous improvement

Delivering poor-quality software on time is not considered success in Agile.

**5. Importance of Organizational Success**

Agile benefits not only software developers but also the entire organization.

**Benefits**

- Faster product delivery
- Better customer satisfaction
- Higher employee productivity
- Increased profits
- Better teamwork
- Continuous innovation

**Example**

Netflix continuously improves its streaming service by quickly adding new features and fixing issues.

**6. Introduction to Agility**

Agility means the ability to respond quickly to changes.

**Characteristics**

- Flexible
- Adaptive
- Fast decision-making
- Customer-focused
- Continuous learning

**Example**

An online shopping website quickly adds new payment methods when customers request them.

**7. How to Be Agile**

Organizations become Agile by following these practices:

1. Deliver software frequently.
2. Communicate regularly.
3. Accept changing requirements.
4. Work in small teams.
5. Improve continuously.

**Example**

Instead of waiting six months to release an app, developers release updates every two weeks.

**Importance of Organizational Success**

Agile helps organizations by:

- Increasing productivity.
- Reducing costs.
- Improving customer satisfaction.
- Delivering products faster.
- Encouraging innovation.

**8. Agile Methods**

Agile methods are different frameworks used to develop software.

**Types**

- Scrum
- Extreme Programming (XP)
- Kanban
- Lean Software Development
- Crystal

**XP focuses mainly on:**

- Coding
- Testing
- Customer involvement
- Continuous feedback

**9. Don't Make Your Own Method**

**Explanation**

Many organizations create their own development process, which often leads to confusion.

Agile recommends using proven methods like Scrum or XP instead of inventing new ones.

**Benefits**

- Saves time
- Reduces mistakes
- Uses industry best practices

**Agile Methods**

Agile methods are different approaches or frameworks used to develop software quickly, flexibly, and with continuous customer feedback. They divide a large project into small iterations, allowing teams to deliver working software frequently.

**Why are Agile methods used?**

Agile methods help teams:

- Respond quickly to changing customer requirements.
- Deliver software faster.
- Improve software quality.
- Increase customer satisfaction.
- Encourage teamwork and communication.

**Types of Agile Methods**

**1. Scrum**

**Definition**

Scrum is the most popular Agile method. It develops software in short time periods called **Sprints** (usually 2–4 weeks).

**Features**

- Product Backlog (list of requirements)
- Sprint Planning
- Daily Scrum Meeting (15 minutes)
- Sprint Review
- Sprint Retrospective

**Real-Time Example**

A shopping app is developed in 2-week sprints:

- Sprint 1: User Login
- Sprint 2: Product Search
- Sprint 3: Payment System

**Advantages**

- Fast delivery
- Better teamwork
- Regular customer feedback

**2. Extreme Programming (XP)**

**Definition**

Extreme Programming (XP) is an Agile method that focuses on **high-quality software**, **continuous testing**, and **customer satisfaction**.

**Main Practices**

- Pair Programming
- Test-Driven Development (TDD)
- Continuous Integration
- Small Releases
- Refactoring
- Customer Involvement

**Real-Time Example**

A banking application is tested after writing every small feature to ensure there are no bugs.

**Advantages**

- High-quality software
- Fewer defects
- Faster bug fixing

**3. Kanban**

Kanban is an Agile method used to **manage workflow** and improve productivity using a visual board.

**Workflow**

To Do → In Progress → Testing → Done

**Features**

- Visual task board
- Continuous work
- Limits work in progress

**Real-Time Example**

A software company tracks all development tasks using a Kanban board.

**Advantages**

- Easy to monitor work
- Reduces delays
- Improves productivity

**4. Lean Software Development**

Lean focuses on **eliminating waste** and delivering **maximum value** to the customer.

**Principles**

- Remove unnecessary work
- Improve quality
- Deliver quickly
- Respect team members
- Continuous improvement

**Real-Time Example**

Develop only the features customers actually need instead of adding unnecessary functions.

**Advantages**

- Lower cost
- Faster development
- Better efficiency

**5. Crystal Method**

Crystal is a flexible Agile method where the development process depends on the **team size** and **project complexity**.

**Features**

- Frequent communication
- Team collaboration
- Simple process
- Customer involvement

**Real-Time Example**

A small startup developing a mobile app uses Crystal because it has only a few developers.

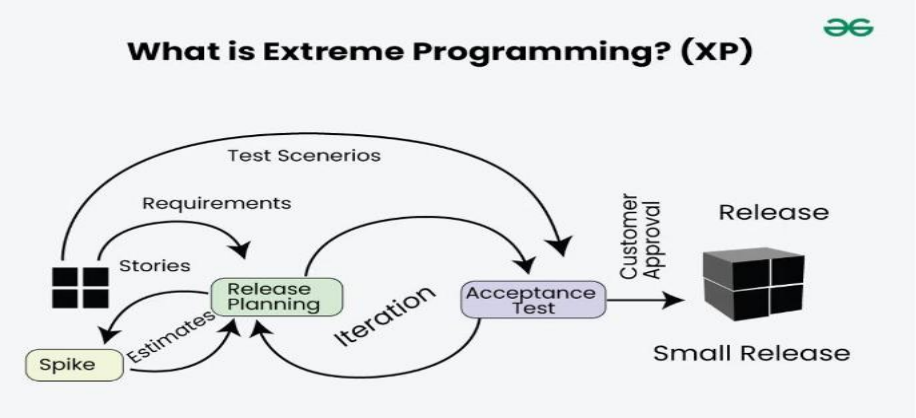
**Advantages**

- Flexible
- Easy to adapt
- Suitable for small teams

**XP Life Cycle**

Steps:

1. Exploration
2. Planning
3. Iterations
4. Production
5. Maintenance
6. Project End




## Implementation of Analysis and Planning in an Agile Application

### 1. Analysis Phase

Analysis is the process of understanding the **customer's requirements** before development starts.

**Steps**

1. Meet the customer to understand the requirements.
2. Gather requirements as **User Stories**.
3. Prioritize the important features.
4. Clarify doubts with the customer.
5. Estimate the effort required.

**Example**

**Application:** Online Shopping App

Customer requirements:

- User Registration
- Login
- Product Search
- Shopping Cart
- Online Payment

These requirements are written as **User Stories**.

**Example User Story:**

"As a customer, I want to search for products so that I can easily find the items I need."

### 2. Planning Phase

Planning is the process of deciding **what work will be done, who will do it, and when it will be completed**.

**Steps**

1. Create the **Product Backlog** (list of all user stories).
2. Prioritize the user stories.
3. Divide work into **Sprints** (2–4 weeks).
4. Assign tasks to team members.
5. Estimate time and resources.
6. Prepare the Sprint Plan.

**Example**

**Sprint 1 (2 weeks):**

- User Registration
- Login

**Sprint 2 (2 weeks):**

- Product Search
- Shopping Cart

**Sprint 3 (2 weeks):**

- Online Payment
- Order Confirmation

At the end of each sprint, the customer reviews the completed features and provides feedback.

**Benefits of Agile Analysis and Planning**

- Better understanding of customer needs.
- Quick response to requirement changes.
- Faster software delivery.
- Reduced project risk.
- Improved software quality.
- Higher customer satisfaction.

**XP Team Roles**

1. **Customer**
    - Defines requirements using user stories.
    - Sets priorities and accepts completed work.
2. **Programmers (Developers)**
    - Write and test the code.
    - Practice pair programming and continuous integration.

3. **Coach**
    - Guides the team in following XP practices.
    - Helps solve process-related issues.
4. **Tracker**
    - Monitors project progress.
    - Compares actual progress with the planned schedule.
5. **Tester**
    - Creates and executes test cases.
    - Ensures the software meets customer requirements.
6. **Consultant**
    - Provides expert advice when specialized knowledge is needed.
7. **Manager**
    - Supports the team by removing obstacles.
    - Ensures resources and coordination are available.

**Adopting XP (How to Implement XP)**

Adopting XP means introducing XP practices into a software project.

**Steps to Adopt XP**

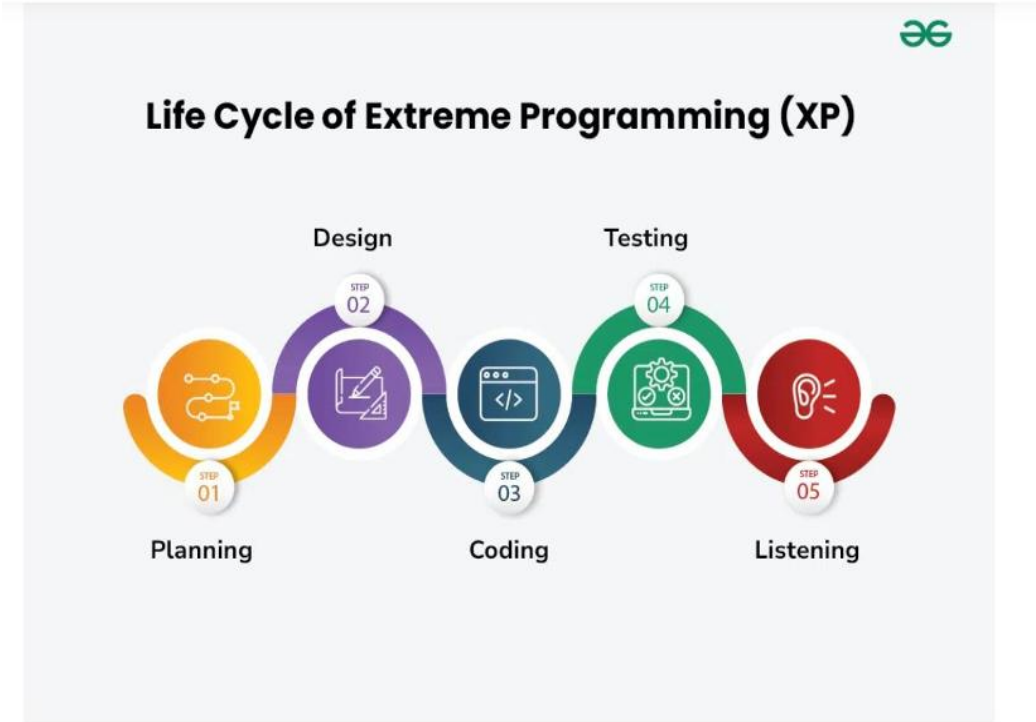
1. Involve the customer throughout the project.
2. Collect requirements as **user stories**.
3. Plan work in short iterations (1–2 weeks).
4. Use **pair programming**.
5. Write **unit tests** before or during coding.
6. Integrate code frequently (**continuous integration**).
7. Refactor code regularly to improve quality.
8. Deliver small, working software releases frequently.

**Benefits of Adopting XP**

- Improves software quality.
- Reduces defects.
- Adapts quickly to changing requirements.
- Increases customer satisfaction.
- Encourages teamwork and faster development.

**Applications**

- **Small projects:** Works well with small teams where face-to-face communication is easy.
- **Projects with new technology or research:** Suitable for projects with frequent changes and technical challenges.
- **Web development projects:** Well-suited for web development projects as the development process is iterative and requires frequent testing to ensure the system meets the requirements.
- **Collaborative projects:** Effective where close interaction between developers and customers is required.
- **Projects with tight deadlines:** Helps deliver results quickly through simple and iterative development.
- **Projects with rapidly changing requirements:** Designed to handle rapidly changing requirements, making it suitable for projects where requirements may change frequently.
- **Quality-focused projects:** Strong emphasis on testing makes it suitable where high quality is essential.



## UNIT–2: Collaborating

**Collaboration** in XP means team members, customers, and stakeholders work together continuously to develop quality software.

**1. Trust**

- Trust means team members believe and support each other.
- Everyone shares knowledge openly.
- It improves communication and teamwork.

**Example:** Developers trust testers to report bugs correctly.

**2. Sit Together**

- All team members work in the same place or communicate continuously online.
- Problems are solved quickly.
- Reduces misunderstandings.

**Example:** Developers and testers sit together to discuss issues immediately.

**3. Real Customer Involvement**

- Customer participates throughout development.
- Gives requirements and regular feedback.
- Ensures the software meets user needs.

**Example:** Bank manager checks every iteration of banking software.

**4. Ubiquitous Language**

- Everyone uses the same simple business terms.
- Developers and customers understand each other.

**Example:** Instead of technical words, everyone says "Transfer Money" instead of "Fund Transaction Module."

**5. Meetings**

XP encourages short daily meetings.

Purpose:

- Discuss completed work.
- Plan today's work.
- Identify problems.

**6. Coding Standards**

- Team follows common coding rules.
- Code becomes easy to read and maintain.
- Reduces errors.

**7. Iteration Demo**

- At the end of every iteration, working software is demonstrated.
- Customer gives feedback.
- Improvements are added in the next iteration.

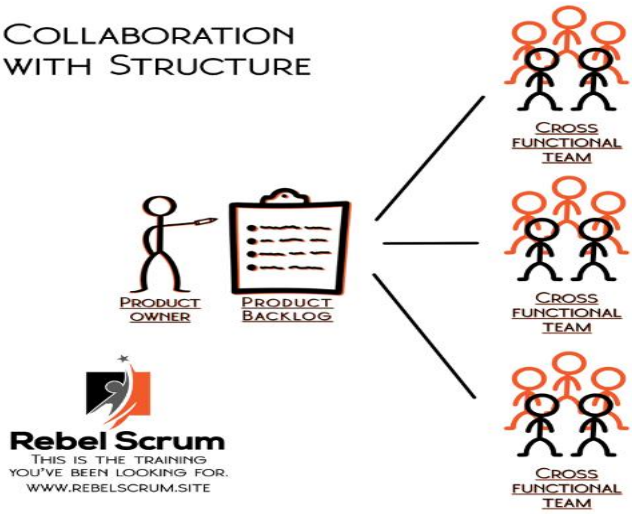
**8. Reporting**

- Progress reports show completed work, pending work, bugs, and future plans.
- Helps management track project status.

**Advantages**

- Better communication
- Faster problem solving
- Customer satisfaction
- High-quality software
- Strong teamwork

**Collaborating process**



## UNIT–3: Releasing

**Releasing** means delivering working software to the customer frequently.

**1. Bug-Free Release**

- Software should contain minimum defects.
- Testing is performed before release.
- Improves customer satisfaction.

**2. Version Control**

- Stores different versions of source code.
- Tracks changes.
- Multiple developers can work together.

**Tools:** Git, GitHub.

**3. Fast Build**

- Software should compile quickly.
- Developers detect errors immediately.
- Saves development time.

**4. Continuous Integration (CI)**

- Developers merge code frequently.
- Every code change is automatically built and tested.
- Detects errors early.

**Benefits :**

- Reduces integration problems.
- Improves software quality.

**5. Collective Ownership**

- Every developer can modify any part of the code.
- No single person owns the project.
- Knowledge is shared among team members.

**6. Documentation**

- XP uses only necessary documentation.
- Focus is on working software.
- Documents include:
    - User stories

- Design notes
- User manuals

**Advantages**

- Frequent software delivery
- Better quality
- Easy maintenance
- Faster development
- Customer satisfaction

## UNIT–4: Planning

Planning in XP means deciding what to develop, when to develop, and how much work can be completed.

**1. Version**

A version is a specific release of software.

Example:

- Version 1.0
- Version 2.0

**2. Release Plan**

Release planning decides:

- Features
- Schedule
- Resources
- Delivery date

Steps:

1. Gather requirements.
2. Prioritize features.
3. Estimate effort.
4. Create release schedule.

**3. Risk Management**

Risk means anything that may delay or affect the project.

Common Risks:

- Requirement changes

- Technical problems
- Lack of resources
- Time delay

Management:

- Identify risks.
- Analyze risks.
- Reduce risks.
- Monitor continuously.

**4. Iteration Planning**

Each iteration usually lasts **1–2 weeks**.

Activities:

- Select user stories.
- Divide work.
- Assign tasks.
- Estimate time.

**5. Slack**

Slack means keeping extra time for unexpected work.

Example:
If work needs 8 days, plan for 10 days.

Benefits:

- Handles unexpected problems.
- Reduces pressure.

**6. User Stories**

User stories describe customer requirements in simple language.

Format:

**As a customer, I want to transfer money so that I can pay bills online.**

**7. Estimation**

Estimation predicts:

- Time
- Cost
- Resources

Methods:

- Story Points
- Planning Poker
- Expert Judgment

**Advantages**

- Better scheduling
- Lower project risk
- On-time delivery
- Efficient resource management

## UNIT–5: Developing

### Introduction

The Developing phase in Extreme Programming (XP) focuses on writing high-quality software through continuous testing, customer feedback, code improvement, and teamwork. XP encourages developers to deliver working software in small increments while maintaining code quality and reducing defects.

The major topics in this unit are:

1. Incremental Requirements
2. Customer Tests
3. Test-Driven Development (TDD)
4. Refactoring
5. Incremental Design and Architecture
6. Spike Solutions
7. Performance Optimization
8. Exploratory Testing

### 1. Incremental Requirements

**Incremental Requirements** means collecting and implementing customer requirements gradually instead of gathering all requirements at the beginning.

In XP, customers may not know all their requirements initially. Therefore, requirements are added step by step during each iteration.

Each iteration develops only a small set of features.

**Steps**

- Customer provides user stories.

- Team selects stories for the current iteration.
- Developers implement the selected stories.
- Customer reviews the software.
- New requirements are added in the next iteration.

**Advantages**

- Easy to manage.
- Flexible to changing requirements.
- Reduces development risk.
- Improves customer satisfaction.

**Example**

A banking application is developed in stages:

- Iteration 1: User Login
- Iteration 2: Balance Enquiry
- Iteration 3: Money Transfer
- Iteration 4: Mini Statement

Thus, the software grows incrementally.

**2. Customer Tests (Acceptance Testing)**

Customer Tests are tests prepared by the customer to verify whether the software satisfies the required business needs.

**Explanation**

The customer defines acceptance criteria before development.

After implementation, developers execute these tests.

If all tests pass, the feature is accepted.

**Purpose**

- Verify customer requirements.
- Ensure correct functionality.
- Improve software quality.

**Advantages**

- Reduces misunderstandings.
- Builds customer confidence.
- Detects errors before release.

**Example**

Customer Requirement:
"I want to transfer money."

Customer Test:

- Enter sender account.
- Enter receiver account.
- Enter amount.
- Click Transfer.
- Money should be transferred successfully.

If successful, the feature passes.

**3. Test-Driven Development (TDD)**

Test-Driven Development (TDD) is an XP practice where developers write test cases before writing the actual program.

**TDD Cycle**

**Step 1: Write Test**

Write a test case for the required functionality.

↓

**Step 2: Write Code**

Write only enough code to pass the test.

↓

**Step 3: Run Test**

Execute the test.

↓

**Step 4: Refactor**

Improve the code without changing functionality.

Repeat this cycle.

**Advantages**

- Fewer bugs.
- Better software quality.
- Easy maintenance.
- High test coverage.

**Real-Time Example**

Requirement:
Calculate ATM withdrawal.

Step 1:
Write a test.

Expected Output:
Balance = ₹10,000

Withdraw ₹2,000

Remaining Balance = ₹8,000

Step 2:
Write the withdrawal program.

Step 3:
Run the test.

Step 4:
Improve the code.

**4. Refactoring**

Refactoring means improving the internal structure of existing code without changing its external behavior.

**Explanation**

After writing code, developers remove unnecessary or duplicate code to make it clean and easy to maintain.

**Common Refactoring Techniques**

- Remove duplicate code.
- Rename variables.
- Simplify methods.
- Improve class structure.
- Split long methods into smaller methods.

**Advantages**

- Improves readability.
- Easier maintenance.
- Reduces complexity.
- Makes future modifications easier.

**Example**

Before Refactoring

total = price + tax + shipping

After Refactoring

total = calculateTotal(price, tax, shipping)

The output remains the same, but the code becomes easier to understand.

**5. Incremental Design and Architecture**

Instead of designing the entire system at once, XP builds and improves the design gradually.

**Explanation**

Initially, only a simple architecture is created.

As new requirements arrive, the design evolves.

This approach avoids unnecessary complexity.

**Benefits**

- Simple design.
- Easy to modify.
- Less maintenance cost.
- Supports changing requirements.

**Example**

Banking System

Iteration 1:
Login Module

Iteration 2:
Account Module

Iteration 3:
Transaction Module

Iteration 4:
Loan Module

Architecture grows step by step.

**6. Spike Solutions**

**Definition**

A Spike Solution is a small experiment or prototype used to investigate a technical problem before implementing the actual solution.

**Explanation**

Sometimes developers are unsure how to implement a feature.

Instead of directly coding the feature, they first create a small prototype.

If successful, they develop the actual feature.

**Advantages**

- Reduces technical risk.
- Improves developer understanding.
- Saves development time.

**Example**

Before integrating a Payment Gateway into an online shopping application, developers first create a small prototype to test whether the payment API works correctly.

If successful, they integrate it into the real application.

**7. Performance Optimization**

Performance Optimization means improving software speed, efficiency, and resource usage.

**Goals**

- Faster execution.
- Lower memory usage.
- Better response time.
- Efficient database operations.

**Methods**

- Optimize algorithms.
- Reduce unnecessary loops.
- Improve database queries.
- Use caching.
- Compress files.
- Remove duplicate code.

**Example**

Before Optimization:
Loading dashboard takes **10 seconds**.

After Optimization:
Loading dashboard takes **2 seconds**.

Customer satisfaction increases.

**Advantages**

- Faster software.
- Better user experience.
- Reduced server load.
- Improved scalability.

**8. Exploratory Testing**

Exploratory Testing is a testing technique where testers explore the software freely without predefined test cases.

**Explanation**

The tester simultaneously:

- Learns the application.
- Designs test cases.
- Executes tests.
- Finds defects.

**Characteristics**

- No fixed test scripts.
- Creative testing.
- Detects hidden bugs.
- Suitable for newly developed applications.

**Example**

Testing an Online Banking Application

The tester:

- Logs in.
- Transfers money.
- Cancels the transaction.
- Refreshes the page.
- Checks whether the balance is updated correctly.

Unexpected bugs may be discovered.

**Advantages**

- Finds hidden defects.
- Improves software quality.
- Increases test coverage.
- Supports continuous testing.

**Advantages of Developing Practices in XP**

- Produces high-quality software.
- Detects bugs early.
- Improves customer satisfaction.

- Supports changing requirements.
- Makes software easy to maintain.
- Reduces development cost.
- Improves teamwork.
- Delivers software quickly.

**Real-Time Example: Online Banking System**

Suppose a team is developing an **Online Banking System** using XP.

- **Incremental Requirements:** First develop Login, then Balance Check, then Fund Transfer, then Bill Payment.
- **Customer Tests:** Bank manager verifies each feature before accepting it.
- **TDD:** Developers write test cases before writing code.
- **Refactoring:** Improve code readability without changing functionality.
- **Incremental Design:** Add new modules one by one instead of designing the entire system at once.
- **Spike Solution:** Test the UPI payment gateway with a prototype before integrating it.
- **Performance Optimization:** Reduce fund transfer time from 8 seconds to 2 seconds.
- **Exploratory Testing:** Testers perform random operations to identify hidden defects.