# Chapter 4: Adopting XP

## Introduction

- XP is powerful but easy to fail
- Most failures happen due to *wrong adoption*, not XP itself
- This chapter helps teams decide:
  - Is XP right for us?
  - Are we ready?

---

## Is XP Right for Us?

XP works best when:

- Requirements change frequently
- Fast feedback is valuable
- Business value matters more than rigid plans
- Team collaboration is possible

XP may struggle when:

- Teams are very large or distributed
- No real customer involvement exists

---

## XP Is a System, Not a Toolbox

- XP practices are *interdependent*
- Picking only a few practices = high risk
- Partial XP is the most common reason for failure

Key Idea:

> Use all practices together

---

## Prerequisite #1 – Management Support

Management must:

- Support frequent releases
- Accept changing requirements
- Trust the team

Without management support:

- XP collapses into waterfall

---

## Prerequisite #2 – Team Agreement

- Entire team must agree to try XP
- XP cannot be forced
- Requires discipline and commitment

Key Risk:

- One non-cooperative member can spoil XP

---

### Prerequisite #3 – Co-located Team

- XP depends on face-to-face communication
- Enables fast feedback and collaboration
- Open workspace preferred

Distributed teams:

- Make XP harder (but not impossible)

---

### Prerequisite #4 – On-Site Customer

- Real business representatives
- Available daily
- Empowered to make decisions

Without on-site customer:

- Team may build the wrong product

---

### Prerequisite #5 – Right Team Size

- Ideal XP team size: 5–12 members
- Too small → lack of skills
- Too large → communication problems

---

## Prerequisite #6 – Use All XP Practices

- XP practices reinforce each other Examples:
- TDD supports refactoring
- Pair programming improves quality
- CI prevents late surprises

Skipping practices weakens the system

---

## Recommendations (Not Mandatory)

Strongly recommended:

- Brand-new codebase
- Strong design skills
- Language easy to refactor
- Experienced XP coach
- Friendly, cohesive team

---

.

### Recommendation #1: A Brand-New Codebase

Easily changed code is vital to XP. If your code is cumbersome to change, you'll have difficulty with XP's technical practices, and that difficulty will spill over into XP's planning practices.

### Recommendation #2: Strong Design Skills

Simple, easily changed design is XP's core enabler. This means at least one person on the team —preferably a natural leader—needs to have strong design skills.

.

### Recommendation #3: A Language That's Easy to Refactor

XP relies on refactoring to continuously improve existing designs, so any language that makes refactoring difficult will make XP difficult. Of the currently popular languages, object-oriented and dynamic languages with garbage collection are the easiest to refactor. C and C++, for example, are more difficult to refactor.

.

### Recommendation #4: An Experienced Programmer-Coach

Some people are natural leaders. They're decisive, but appreciate others' views; competent, but respectful of others' abilities.

XP relies on self-organizing teams . This kind of team doesn't have a predefined hierarchy; instead, the team decides for itself who is in charge of what. These roles are usually informal.

If you have no obvious coach… Explain the situation to the team and ask them to choose a coach by consensus.

.

### Recommendation #5: A Friendly and Cohesive Team

XP requires that everybody work together to meet team goals. There's no provision for someone to work in isolation, so it's best if team members enjoy working together

---

## Starting XP – "Go!"

- Start once prerequisites are met
- Follow practices strictly
- Expect discomfort initially

Important:

- XP exposes problems early

## The Challenge of Change

Common resistance:

- Fear of less documentation
- Fear of no upfront design
- Resistance to pair programming

Reality:

- XP reveals hidden problems

---

## Applying XP to a New Project

Recommended approach:

- Start XP on a brand-new project
- Easier mindset shift
- Faster success

---

- First 2-3 weeks the on-site customer will work out the release plan. Programmer establish the technical infrastructure and other learning how to work together.
- First activity: selecting stories from release plan .Think of one feature that will definitely be the part of first release .
- Next the programmer should start establishing technical infrastructure. Set up an integration machine , create version control repository.
- During first iteration its good all the programmers work on first few stories as a group.
- Break into pairs and work on separate parts without interfiering with others.

## Applying XP to an Existing Project

Challenges:

- Legacy code
- Existing habits
- Missing tests

Strategy:

- Add tests first
- Improve incrementally

---

## XP in Phase-Based Organizations

- Organization may remain phase-based
- XP team works iteratively internally
- Requires negotiation with management

---

## Extremities – Using Bits of XP

Warning:

- Using only stand-ups or pairing is risky
- Partial XP leads to poor results

Conclusion:

- Either adopt XP seriously or don't

---

## Key Takeaways

- XP adoption requires organizational change
- Management and team agreement are critical
- XP works best as a complete system
- Partial XP is dangerous