

Chapter 5: Thinking

Anil

Why “Thinking” Matters in XP

- XP is not just practices, but mindset
- Thinking guides day-to-day decisions
- Helps teams improve quality and collaboration

Key Idea:

Good thinking leads to good software

Contents

XP doesn't require experts. It does require a habit of mindfulness.

- **Pair Programming:** Pair programming doubles the brainpower available during coding, and gives one person in each pair the opportunity to think about strategic, long-term issues.
-
- **Energized Work:** Energized work acknowledges that developers do their best, most productive work when they're energized and motivated.
 - **Informative Workspace:** An informative workspace gives the whole team more opportunities to notice what's working well and what isn't.
 - **Root-Cause Analysis:** Root-cause analysis is a useful tool for identifying the underlying causes of your problems.

- **Retrospectives:** Retrospectives provide a way to analyze and improve the entire development process.
-

Pair Programming – What It Is

Pair programming refer to two people working at the same keyboard one person codes—the driver. The other person is the navigator, whose job is to think.

- Two programmers, one computer
- Driver: writes code
- Navigator: reviews and thinks ahead

Goal:

- Better design and fewer defects
-

Why Pair Programming Works

- Continuous code review
- Knowledge sharing
- Better focus and discipline
- Fewer bugs and rework

Common Myth:

- Pairing wastes time
-

How to Practice Pair Programming

- Switch roles frequently
 - Pair with different people
 - Pair all production code
 - Use pairing stations
-

Challenges in Pair Programming

- Personality conflicts
- Fatigue
- Resistance from individuals

XP Solution:

- Rotate pairs
 - Take breaks
 - Build trust
-

Energized Work – Core Idea

- Sustainable pace
- No overtime heroics
- Quality suffers when tired

XP Belief:

Tired developers make bad decisions

How to Maintain Energized Work

XP's practice of energized work recognized that although professionals do work under difficult circumstances, they do their best productive work when they are energized and motivated.

Go home on time. Stay home when you are sick. Having healthy food available at workplace. Taking breaks. Extended overtimes will not solve the scheduled problems.

- Work normal hours
 - Take regular breaks
 - Respect work-life balance
 - Avoid burnout
-

Result: When the team is energized as a group you pay attention to detail and look for opportunities to improve work habits.

Contradictions: Energize work is not an excuse to go off. Generate trust by putting in fair day's work.

Informative Workspace

- Big Visible charts
- Hand-drawn charts
- Process improvement charts

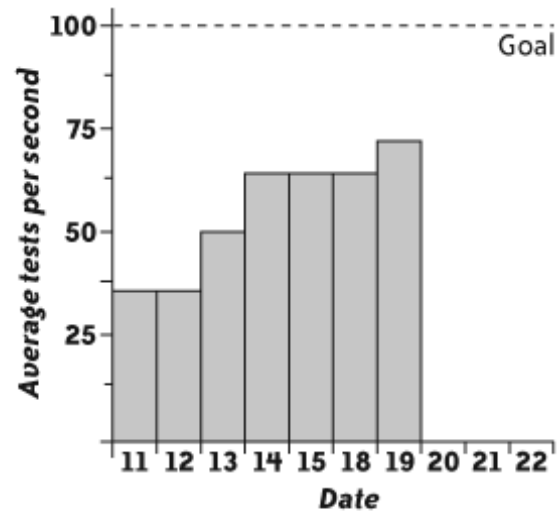
Result: If team has an informative workspace, they will have up-to-date information about all the important issues in the team.

Contradictions: If team does not sit together in a shared workspace, they probably won't be able to create an effective informative work



MO						
JS						
SW	✓	✓				
NS		✓				
MV				✓		
SS		✓		✓	✓	
	MO	JS	SW	NS	MV	SS

(a) Pair combinations



(b) Tests per second

Informative Workspace – Meaning

- Workspace that shows project status
- Information is visible to everyone

Examples:

- Story cards
 - Task boards
 - Charts
-

Benefits of Informative Workspace

- Reduces need for meetings
 - Improves transparency
 - Helps quick decision-making
-

Root-Cause Analysis – Purpose

- Fix problems permanently
- Not just symptoms

XP Question:

Why did this problem happen?

Classic approach to root-cause analysis is to ask “why” five times...

Result: When root- cause analysis is instinctive reaction your team values fixing problems rather than placing blame

Contradictions: every problem has a cause outside the control, but don’t use this as an excuse.

A mistake-proof process is neither achievable nor desirable.

How to Do Root-Cause Analysis

- Identify the problem
 - Ask “Why?” repeatedly
 - Find the real cause
 - Fix the process
-

When NOT to Fix Root Cause

- Fix is too expensive
 - Problem is rare
 - Temporary workaround is enough
-

Retrospectives – What They Are

- Regular reflection meetings
- Team discusses what went well and what didn't

Goal:

- Continuous improvement
-

Types of Retrospectives

- Iteration retrospective
 - Project retrospective
 - Ad-hoc retrospectives
-

Iteration Retrospective – Steps

1. Prime Directive (no blame)
 2. Brainstorming
 3. Mute mapping
 4. Choose improvement action
-

After the Retrospective

- Implement improvements
 - Track results
 - Repeat regularly
-

Results of Good Thinking Practices

- Higher quality software
 - Better teamwork
 - Fewer surprises
 - Continuous learning
-

Contraindications

- Pairing may not suit all tasks
 - Energized work needs discipline
 - Retrospectives fail without honesty
-

Alternatives

- Code reviews instead of pairing
 - Status reports instead of boards
 - Informal discussions instead of retrospectives
-

Key Takeaways

- XP thinking practices support technical practices
- Reflection is essential for agility
- Improvement is continuous