unit 2

anil

Chapter 6: Collaborating

Team & Organizational Strategies

Trust

"Agile succeeds not because of tools or techniques, but because people choose to collaborate with empathy, honesty, and responsibility."

Strategy Type	Core Idea
Team Strategies	Build empathy & trust inside the team
Organizational Strategies	Build credibility & support outside the team

---

Why Collaboration Matters in Agile

- Technical skills alone are not enough
- Most project failures are people-related
- Agile success depends on collaboration

Key Idea:

> Agile is about people working together effectively

Team Strategies vs Organizational Strategies

- Team Strategies: Inside the team
- Organizational Strategies: Outside the team
- Both are required for Agile success

Team Strategy #1 – Customer–Programmer Empathy

- Programmers think in logic
- Customers think in business value
- Empathy bridges this gap

Definition:

> Understanding customer problems, not just requirements

Why Customer–Programmer Empathy Matters

- Prevents building wrong features
- Improves customer satisfaction
- Encourages shared ownership

Team Strategy #2 – Programmer–Tester Empathy

- Testers are partners, not enemies
- Bugs are feedback, not criticism

Definition:

> Mutual respect between programmers and testers

Benefits of Programmer–Tester Empathy

- Higher quality software
- Fewer conflicts
- Faster feedback loops

Team Strategy #3 – Eat Together

- Informal communication builds trust
- Reduces fear and hierarchy
- Encourages open discussion

Why Eating Together Works

- Conversations happen naturally
- Problems surface early
- Team bonding improves

Team Strategy #4 – Team Continuity

- Keep the same team together
- Avoid frequent member changes

Definition:

Stable teams perform better over time

Benefits of Team Continuity

- Knowledge retention
- Increased trust
- Predictable performance

Impressions

- Teams are judged by behavior
- Reliability and attitude matter
- Impressions influence support

Organizational Strategies

- How teams interact with management
- How teams earn trust and support

Organizational Strategy #1 – Show Some Hustle

- Be proactive
- Respond quickly to issues
- Visible effort builds confidence

Organizational Strategy #2 – Deliver on Commitments

- Make realistic promises
- Meet deadlines consistently

Key Idea:

Trust is built by keeping commitments

Organizational Strategy #3 – Manage Problems

- Identify problems early
- Communicate issues openly
- Fix root causes

Organizational Strategy #4 – Respect Customer Goals

- Customers care about outcomes
- Business value over technical elegance

Organizational Strategy #5 – Promote the Team

- Share successes
- Make work visible
- Gain organizational support

Organizational Strategy #6 – Be Honest

- Share accurate information
- Bad news early is better than late

Definition:

> Honesty builds long-term credibility

Summary – Team Strategies

- Empathy builds trust
- Informal interaction matters
- Stable teams perform better

Sit Together

Why Communication Fails

- Delays in responses
- Misunderstandings
- Too many handoffs
- Overdependence on tools

Key Idea:

> Distance increases misunderstanding

Sit Together – Core Concept

- Team members work in the same physical space
- Communication becomes immediate
- Problems surface early

Definition:

> Sit Together means colocating the whole team in one shared workspace

---

Why XP Emphasizes Sitting Together

- Face-to-face communication is fastest
- Reduces documentation overhead
- Encourages spontaneous discussion

Accommodating Poor Communication

- Some teams communicate poorly at first
- XP does not blame people
- Environment is adjusted to help

Examples:

- Closer seating
- Clear boards
- Short conversations

Problems with Poor Communication

- Hidden assumptions
- Rework and defects
- Delayed decisions

XP Insight:

> Fix communication problems, not people

A Better Way

- Instead of emails and tickets:
  - Talk directly
  - Clarify immediately

XP Belief:

> Communication works best when it is human

Exploiting Great Communication

- Strong teams benefit even more from sitting together
- Faster decision-making
- Shared understanding

Benefits of Exploiting Great Communication

- Higher productivity
- Better design decisions
- Stronger team trust

Secrets of Sitting Together

- Sit close enough to talk easily
- Avoid physical barriers
- Maintain eye contact

What Sitting Together Is NOT

- Not about supervision
- Not micromanagement
- Not removing privacy completely

Making Room

- Management must support workspace change
- Remove cubicles and walls
- Provide shared tables

XP Message:

> Workspace design affects team behavior

Designing Your Workspace

- Large shared tables
- Whiteboards nearby
- Visible task boards
- Comfortable seating

Design Goals of XP Workspace

- Easy communication
- High visibility
- Team ownership

Sample Workspaces

- Open tables
- Wall-mounted boards
- Shared screens

Purpose:

> Anyone can understand project status quickly

What Sample Workspaces Show

- Work in progress is visible
- Collaboration is natural
- Less need for meetings

Adopting an Open Workspace

- Gradual transition recommended
- Address concerns early
- Involve the team in design

Common Concerns About Open Workspaces

- Noise
- Distractions
- Loss of privacy

XP Response:

- Establish team norms
- Quiet zones when needed

Benefits of Open Workspace

- Faster feedback
- Stronger collaboration
- Better problem-solving

Ubiquitous Language & Domain Collaboration

Agile / Extreme Programming (XP)

Why Language Matters in Software Development

- Most software failures are **not coding problems**
- They are **communication problems**
- Same word → different meanings
- Agile focuses on **shared understanding**

If we misunderstand the problem, we will build the wrong solution.

What Is Ubiquitous Language?

Definition

- A **shared vocabulary**
- Used by:
  - Customers
  - Developers
  - Testers
  - Product managers
- Used **everywhere**:
  - Conversations
  - Documents
  - Tests
  - Code

One language, no translation

Simple Example – Language Confusion

Customer says:

- “The order is shipped”

Developer understands:

- Packed in warehouse

Customer means:

- Handed over to courier

Result: Bug without a code error

Solution: Shared language

The Domain Expertise Conundrum

The Core Problem

- Customers know the **business**
- Developers know the **technology**
- Neither knows everything

Role	Knows	Doesn't Know
Customer	Business rules	Software design
Developer	Code	Business nuances

Gap causes misunderstanding

Two Languages in Traditional Development

Business Language

- Orders
- Discounts
- Inventory
- Policies

Technical Language

- Classes
- Databases
- APIs
- Algorithms

Translation causes errors

Documents become outdated

Agile Solution – Speak the Same Language

Agile encourages:

- Continuous conversation
- On-site customer
- Frequent feedback
- Removing translators

One shared domain language

Faster decisions

Fewer defects

How to Build a Shared Language

Agile practices that help:

- Customer involvement
- Pair programming
- Iteration demos
- Customer tests as examples
- Daily communication

Language grows through collaboration

Ubiquitous Language in Code

Agile rule:

> "If it matters to the business, it belongs in the code."

Poor naming:

```
processData()
calculateX()
```

Domain naming:

```
allocateInventory()
calculateShippingCost()
```

Code becomes readable to non-programmers

Refining the Ubiquitous Language

Language is **not fixed**

- Understanding improves over time
- Terms are refined
- Code and tests are updated

Example:

- “Priority Customer” → “Gold Customer” → “Platinum Customer”

Refactoring applies to language too

The Role of Questions in Agile

Agile encourages **asking questions**

- “What does this term mean?”
- “When exactly does this happen?”
- “Are these two terms the same?”

Questions prevent defects

Silence creates bugs

Healthy Questions Improve Understanding

Good teams:

- Ask early
- Ask often
- Ask without fear

Agile culture:

- No blame
- High trust
- Shared responsibility

Summary – Key Takeaways

- Ubiquitous Language = shared vocabulary
- Solves domain expertise gap
- Removes translation errors
- Used in conversation, tests, and code
- Continuously refined
- Supported by questions and collaboration