

XP

Extreme Programming



XP – An Agile Technique

- A different type of software process
- Feedback oriented
- Origin
 - Kent Beck
 - Martin Fowler

DeMarco comments

- Process versus flexibility – armor versus mobility
- Most important new development in software engineering?

Evolutionary development basic Idea

- Ready fire aim
 - Coarse aim – redesign with feedback
- Driving car analogy
 - Use feedback to make small corrections
- Develop ability to make small changes
corrections

Changing code

- Must be able to change code
 - Redesign – refactor

Changing directions

- Must be able to make small corrections in direction of project
- Scope
 - Variables - Cost, Time Quality, Scope –
 - focus on scope
 - Stories – use cases

Ease of change

- Lightweight in order to make easy corrections
 - Simple design
 - Simple tools
- Small team 10- 20

Practices

- Planning Game
- Small releases
- Metaphor
- Simple design
- Testing
- Refactoring

Practices (cont.)

- 40-hour week
- On-site customer
- Coding standards
- Pair programming
- Collective ownership
- Continuous integration

Facilities

- Open workspace
- Small private spaces around periphery
- Common programming area in middle

Scoping a project

- Does the project make sense
- Big stories
- Rough estimates of the time to implement each
- Budget
- Constraints

Planning Game

- Players
 - Development
 - Business
- Pieces – story cards
- Phases
 - Exploration
 - Commitment
 - Steering

Pieces (story cards)

- Understandable to customers and developers
 - sentence or two – index card
 - agreement to talk about feature
- Testable
- Valuable to the customer
- Small enough so that programmers can build a few in each iteration (each iteration 2-3 weeks)

Exploration Phase

- Write a story
- Estimate a story
- Split a story -if needed

Commitment Phase

- Sort by value
- Sort by risk
- Set velocity
- Choose scope

Steering Phase

- Iteration
- Recovery
- New story

Iteration planning game

- Players programmers
- Pieces are task cards
 - something that can be done in a few days
- Moves
 - Exploration Phase
 - Commitment Phase
 - Steering Phase

Development Strategy

- Continuous Integration
 - after a few hours (no more than a day) integrate
 - must have fast build time
 - reasonably complete test suite that runs in a few minutes
 - collisions
 - low chance two pairs of programmers change same class or method at same time
 - reconcile easy – represents few hours of work

Development Strategy

- Collective ownership
 - anyone can change any piece of code in the system at any time
 - tests save you
 - others simplify your complex code
 - feeling of personal power
 - not stuck with someone else's stupidity
 - spreads knowledge of system

Development Strategy

- Pair programming
 - dialog not just one person sitting and watching
 - switch pairs – my task your task
 - information spreads through group
 - code quality higher
 - maintain standards testing etc.
 - you learn to communicate about code

Design strategy

- Simplest thing that could possibly work
 - simplest design easiest to communicate
 - feedback – know when design right and wrong quickly
 - need simple strategy
- Problem with design up front
 - cost of up front design
 - may not need features for tomorrow
 - may learn better how to design between now and then

Design strategy

- THE STRATEGY

- Start with a test. Forces some design just to write test. What are the objects and their visible methods?
- Design and implement just enough to get that test running. You will have to design enough of the implementation to get this test and all previous tests running.
- Repeat.
- If you ever see the chance to make the design simpler, do it.

Testing Strategy

- General
 - Tests automatic
 - tests isolated
 - test things that might break
- Programmer tests
- Customers write tests story-by story
- Other tests

Can we adapt XP?

- Problems with size
- Problems with customers
- Problems with programmers