Extreme Programming:
So Crazy It Just Might Work

Robert Penner

FITC 2005

Some Risks in Software Development

- Schedule slips
- Project canceled
- System calcifies
- Too many defects
- Business misunderstood
- Business changes
- False feature rich
- Staff turnover

Stereotypical Approaches to Development

- Big Plan Up Front (waterfall)
- Seat of Your Pants (cowboy)
- XP literature addresses Big Plan Up Front mostly
- XP aims to increase both discipline and flexibility

XP Practices

- The Planning Game
- Small Releases
- Metaphor
- Simple Design
- Testing
- Refactoring
- Pair Programming
- Collective Ownership
- Continuous Integration
- Forty-Hour Week
- On-Site Customer
- Coding Standards

My Current Project

- The MusicRain Viewer (demo)

The Planning Game

- Eisenhower: In preparing for battle I have always found that plans are useless, but planning is indispensable.
- Stories
- Iterations
- Business vs. Technical Decisions

Small Releases

- Release early, release often
- The application grows organically

Metaphor

- A shared understanding of the system
- “Desktop”
- “Files”
- “Folders”

Testing

- I lost my wallet recently
- How long ago did the bug enter the system?
- How do you know something broke because of a change?
- Testing thoroughly and often is crucial but boring

Unit Tests

- Have confidence
- Have fun
- Tests serve as documentation that is always up to date

Test-Driven Development Cycle

- 1. Write a test
- 2. Write just enough code to pass the test
- 3. Run the test suite
- 4. If any tests fail, go to #2
- 5. Refactor
- 5. Repeat

Acceptance Tests

- Acceptance Tests verify a working system from the customer's point of view
- The customer writes them (with help)

Simple Design

- “The simplest thing that could possibly work”
- “You aren’t going to need It”
- “Make it work, make it right, make it fast”
- Helps procrastinators get started

Refactoring

- Refactoring is restructuring code without changing its overall behavior
- Supported by unit tests
- The Big Ball of Mud
- The Pasta Theory of Code
- Smells
- You get good at changing because you're doing it all the time

Pair Programming

- Two developers at one machine working on the same code
- Driver and Navigator
- Pair Pressure
- Your Truck Number

My Experience with Pair Programming

- Effective and fun
- Skype for voice
- Screen Sharing: Windows XP Remote Assistance (fast)
- Or TightVNC (cross-platform)

Costs and Benefits of Pair Programming

- A Study by Laurie Williams
- Total development time increased by 15% (but not 100%)
- Defect rate decreased by 15%
- Mistakes are caught sooner
- Designs are better with less code
- Problems are solved faster by tag-teaming
- Improved team dynamics and communication
- Peer mentoring
- People enjoy their work more

Collective Ownership

- Anyone on the team can change any code at any time, as long as the tests pass
- Trend: Open Source, Wikipedia
- Benefit: Agility, Speedy updates

Continuous Integration

- Integrate and build the system many times a day
- All the tests have to pass
- Easier to find a bug's point of entry

Forty-Hour Week

- The number of hours can vary, but the pace needs to be sustainable
- Respect the humanity of the software development process
- Don’t work overtime a second week in a row

On-site Customer

- An end user is available to the team full-time
- Answers questions, sets priorities
- Expensive to do, expensive not to do

Coding Standards

- Naming Conventions
- Formatting, indentation
- Curly Brace Wars
- Makes pair programming and collective code ownership easier

Case Studies

- Escrow.com Migrates to Extreme Programming
- Laurie Williams: Pair Programming

Adopting XP

- Pick your worst problem
- Apply XP to it
- Unit testing and pair programming are good to try

Resources

- robertpenner.com/presentations/
- XP Books by Kent Beck
- extremeprogramming.org
- xprogramming.com
- pairprogramming.com
- asunit.org