

Introduction to XP values

Slides from XpDay3 plus some slides explaining more
about XP practices



Introduction to XP

- Duncan Pierce will talk about Agile values
- Sean Hanly will talk about XP's process
- Hopefully plenty of time for questions

What are values?

- Easy to concentrate on practices like automated testing
- Values are more subtle
- Underlying principles that guide everything in the process

XP's values

- *Courage* - making commitments, sharing responsibility, not fearing uncertainty
- *Simplicity* - incremental investment, maintaining understandability
- *Feedback* - making early returns, measuring
- *Communication* - making the information that matters visible

Counter-intuitive

- Involve changing “thinking habits”
- This takes time and effort
- But agile thinking is more fundamental than agile processes

Approaching agility

- Making incremental investments
- Getting feedback
- Understanding and managing risks

Approaching agility

- Looking for early returns
- Changing one thing at a time
- Doing things simply

Approaching agility

- Looking for “good enough” not “perfect”
- Believing that organic, non-mechanical (chaordic) processes can work
- Not being afraid of information about progress

Migrating to agile

- Coming away from the conference all fired up for a big process change
- Easy to throw away what you have that's good
- Better to make lots of small process changes...
- Leading in the direction of a larger vision

What are values for?

- Values can guide decisions which are otherwise not clear-cut
- Early on, it's worth considering decisions against values like XP's

The end

- To take away:
 - try to avoid concentrating solely on practices
 - look for what is being achieved
 - think about agile processes in terms of values

xp practice

Collective ownership

- Nobody owns any part of the code
- Anyone can work on anything
- Anyone's code can be changed / deleted by anyone else

xp practice

On-site customer

- Customer is considered to be an essential part of the team in XP
- Customer is a domain expert
- Always available ("on site") to answer queries or make small-scale decisions
- Takes business and scheduling decisions
- May act on behalf of many users of system

Stories

- Customers or users write stories
- Each story captures one requirement scenario
- Typically recorded on standard index card for ease of management

Front

A story card

Short descriptive
title

Cup of Tea

thing wanted

I want *a cup of tea*
so that *I feel refreshed*

business benefit
expected

Developers' estimate — *5 minutes*

Back

Acceptance tests

Action

Put thermometer in cup

Drink liquid in cup

Result

Thermometer shows
over 80 celsius

Tastes like tea

xp practice

Planning game

- Planning happens once at the start of each iteration
- Plan for just one iteration at a time
- Planning game is the main interface between customer and development team

How it works

- Customer provides all the story cards that have been written
- Developers write on each card an estimate of how long it will take to implement
- Developers provide an estimate of available developers \times time (= “velocity”)
- Customer chooses stories by priority to fill the available time

Load factor

- Load factor is the multiplier that connects how long you think things will take to how long they actually take
- e.g. you think 2 days, it takes 6, so load factor is 3 (which is fairly typical!)
- You probably don't need load factor calcs in this workshop because time units are small

Ideal time

- When load factor is in use, estimate in “ideal time” e.g. ideal days
- Divide velocity by load factor
- Commit to this amount of work per iteration
- After iteration, divide real time spent by ideal time estimated to get new load factor

xp practice

Pair programming

- You probably noticed there's only half as many computers as people...
- XP advises pair programming

Advantages

- Problems are solved much faster by 2 people
- Most trivial but annoying mistakes are caught immediately
- People learn from their partner
 - coding techniques
 - new parts of the system

Roles in a pair

- Typically, programmers take it in turns
- One codes or writes tests
- The other watches for mistakes, gives advice or thinks about the bigger picture
- They often talk about what they are working on and the next step they will take
- Is this what you're experiencing?

xp practice

Automated testing

- Testing is important in ensuring software is reasonably reliable
- If you release frequently a manual testing phase is a serious overhead
- In some scenarios it is difficult to reproduce test cases by hand
- Better to automate your tests and run them regularly (even almost continuously)

xp practice

Simple design

- What's the simplest thing that could possibly work?
- 90% of the time this is all you need
- It is easier to add complexity than to take it away
- Problems become more clear-cut when you already have a partial solution

xp practice

Coding standards

- The code is the only guaranteed-correct documentation for a system
- You may have other documents and comments, but:
 - these can get out of date
 - they can become misleading

xp practice

Refactoring

- Changing the way code is written / structured without changing what it does
- Enables
 - readability improvements
 - restructuring for testing
 - just-in-time design
- Use tests to back up refactoring steps

xp practice

40-hour week

- Don't work more than a 40-hour week
- Long hours make more productivity only for a very short time
- System quality suffers a lot through long hours over long periods
- In XP, you go fast by maintaining the quality & fluidity of system

Practice interactions

