CSE70: Lecture 2

• Read the class web page
• First customer meeting today
  – New milestone
• No tutor
• On-line feedback form
• Today
  – Software process, XP, and Pair Programming
  – More on project…
Software Development Process

• Defines
  – Phases, stages, methods, techniques, practices,…
• Software Process
• Development Process
• Why do you need a process?
  – Understanding, standardization, quality
  – Organization and support
  – Manage complexity, quality, and costs
The Waterfall

- First development model (1970s)
- Serial approach with Big Design Up Front (BDUF)
The Waterfall

• Disadvantages
  – Initial phases are based on partial information
  – Hard to adapt and improve the result of earlier phases

• Never use a waterfall approach?
  – Depends
    • E.g. …?
Iterative Development

• 1980s
  – Develop quickly and cheaply
• Spiral model (Boehm 1988)
  – The waterfall in a “loop”?  
  – Reduces risks
    • ???
    • Eliminating errors early
    • Finding better alternatives
Iterative Development

• Advantages
  – Recognizes the need for reiterating phases
  – Suited for
    • incomplete requirements
    • rapidly changing requirements

• Disadvantages
  – Complexity
  – (Hard to define deliverables?)
Incremental Development

• Smaller releases
  – No “big-bang” release
  – Sequence of *product baselines*

• Advantages
  – Product delivered sooner
  – Enable early changes
  – Less expensive and less risky

• Not always adequate
eXtreme Programming

• 4 Values
  – Simplicity
  – Communication
  – Feedback
  – Courage

• 13 Practices
  – Whole Team
  – Planning Game, Small Releases, Customer Tests
  – Simple Design, Pair Programming, Test-Driven Development, Design Improvement
  – Continuous Integration, Collective Code Ownership, Coding Standard
  – Metaphor, Sustainable Pace
XP Practices

• Whole Team (On-Site Customer)
  – To steer the project

• Planning Game
  – Release planning
  – Iteration planning

• Small Releases
  – Functional products

• Customer Tests
XP Practices

• Pair Programming
• Test-Driven Development
  – Test first
  – Test always
• Simple Design
  – Is there a lot or a little of design in XP?
• Design Improvement (Refactoring)
XP Practices

- Continuous Integration
  - Late integration is problematic
- Collective Code Ownership
- Coding Standard
- Metaphor
  - Common vision of the system
- Sustainable Pace (40 Hours Week)
# XP: values, practices, and tools

<table>
<thead>
<tr>
<th></th>
<th>SVN</th>
<th>Ant</th>
<th>JUnit</th>
<th>Concordion</th>
<th>Trac</th>
<th>Simplicity</th>
<th>Communication</th>
<th>Feedback</th>
<th>Courage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Whole team</td>
<td>X</td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Planning game</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Small Releases</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Customer Tests</td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Simple Design</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Pair Programming</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Test Driven Development</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Design Improvement</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Continuous Integration</td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Collective Code Ownership</td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Coding Standard</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>Metaphor</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sustainable Pace</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>
eXtreme Programming?

- What is *extreme* about it?
- Is it always appropriate?
- Are we doing XP in the class project?
Pair Programming

• Collaborative effort
• Why not doing?
  – Requires more resources
  – Programming is considered a solitary activity
• Why doing?
  – Faster
  – Produces better design
  – Gains from partners contribution
Aspects of Pair Programming

• Results (by Cockburn & Williams)
  – Economics
  – Satisfaction
  – Design quality
  – Continuous reviews
  – Problem solving
  – Learning
  – Team building and communication
  – Staff and project management
The Economics of Pair Programming

• Increases development cost
  – 15% more programming time

• Decrease in defects
  – 15% less defects

• 50K LOC at a 50LOC/h rate
  – 1000 hours – 1150 hours

• .1 defects/LOC (only 70% are removed)
  – 1500 defects – 1275 defects

• 10 hours/defect
  – 15000 hours – 12750 hours
More benefits from Pair Programming

• Satisfaction
  – Statistically more enjoyed

• Design Quality
  – Combination of different background, information, perspective
  – Reduction of lines of code

• Continuous reviews
  – Immediate and continuous inspection
  – Better accepted than inspection
  – More attention to standards (and less slack?)
More benefits from Pair Programming

• Problem Solving
  – Pair-relaying

• Learning
  – Line of sight
  – Expert in Earshot

• Team Building and Communication

• Staff and Project Management
  – Improved staff skills
  – Reduced staff risk
Summary

• Software Process (reader)
  – Serial, iterative, and incremental
• XP (paper by Ron Jeffries)
  – 4 values, 13 practices
• PP (paper by Cockburn & Williams)
  – statistical data and benefits

• For the next time…
Questions about the project?

• First assignments?
• Typical requirements for IRC commands
  – help
  – invoke
  – Replies from server?
• Will you do a bot or a graphical client?
• What about testing?