XP in an Hour

Joe Bergin & Fred Grossman
Computer Science and Information Systems
Pace University
Introduction

• Extreme Programming is a high discipline and very iterative development method

• XP avoids early commitment and early infrastructure development to achieve:
  • Low cost of change and
  • Easy retargeting of a project
Why Projects Fail

• Trying to over-control the dimensions
  • Features
  • Cost
  • Schedule
• Quality
• In reality you can only control 3 of these
Values in XP

- Courage
- Communication
- Simplicity
- Feedback
- Respect
Key Ideas

- Practices are synergistic & support each other
- Distance is expensive
- Schedules never slip
- Balance between rights & responsibilities
- Set of practices is humane
What It Gives You

• Rights AND Responsibilities
• Humane work environment
• Skills that are valuable
• Pride of workmanship
What It Requires

• Discipline
• Commitment
• Honesty
• Courage
What is missing?

• Upfront requirements gathering and sign-off -- hence no need to commit early

• Upfront design documents -- hence easy to retarget

• Early costs amortized over life of project

• Intimidation: schedule, cost, or value
Roles: Customer

• Write short “story cards” describing features

• Answer questions throughout to add specificity to the stories

• Write acceptance tests to verify stories

• Make all business decisions: function, priority, feature value, acceptance
Sample Story

Story 3

• The system will correctly classify triangles: right triangles, equilateral, etc.
Sample Acceptance Test

Task 3.1 (Part of story 3)

Write a function named right that will take three inputs representing the sides of a triangle and return whether that is a right triangle or not.

<table>
<thead>
<tr>
<th>myFixtures.rightTriangle</th>
<th>a</th>
<th>b</th>
<th>c</th>
<th>right()</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>true</td>
</tr>
<tr>
<td>b</td>
<td>6</td>
<td>8</td>
<td>10</td>
<td>true</td>
</tr>
<tr>
<td>c</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>false</td>
</tr>
<tr>
<td>d</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>false</td>
</tr>
</tbody>
</table>

These are created in Excel or HTML, but are executable
After Execution

Task 3.1 (Part of story 3)

Write a function named right that will take three inputs representing the sides of a triangle and return whether that is a right triangle or not.

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>right()</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>5</td>
<td>true</td>
</tr>
<tr>
<td>6</td>
<td>8</td>
<td>10</td>
<td>true</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>9</td>
<td>false</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>7</td>
<td>false</td>
</tr>
</tbody>
</table>

Failed tests show up in red.
Roles: Developer

• Estimate stories
• Break stories into tasks
• Build tasks -- with customer feedback
• Write unit tests (all tests always succeed)
• Do continuous integration
Roles: Other

• Tracker (keep everyone aware of progress)
• Coach (conscience of the team)
• Big-Boss (management and shelter)
• Tester (write/run unit tests...)
• Consultant (extra knowledge as needed)
Contract

• For best effort and full communication, NOT for deliverables on a given date

• Customer may terminate project at any time

• Short release cycles (4-6 weeks) ensure constant delivery of customer value

• Schedule never slips, though features may be dropped from an iteration
Controlling Cost

• Build the high value features first -- controlled by customer

• Make expensive decisions as late as possible

• When the cost and value curves cross - quit!
Staying Happy

• Customer steers like a bicycle

• If something is not “right” then write a new story and prioritize it like any other

• Developers build only the stories in the current iteration and always do the simplest thing that could possibly work

• Stories are fine-grained to enable short iterations
Practices

• XP has a dozen or so key practices. The most important overall are
  • Onsite Customer
  • Whole Team
The most important practices for the customer are:

- Onsite customer - available customer
- Planning Game
- Customer Written Acceptance Tests
Onsite Customer

• Customer is needed on site because
• Developers should not make business decisions but
  • no upfront requirements
  • no upfront design documents
• A story is a contract to talk in the future
Whole Team

• In addition to the customer, the “whole team” includes all personnel with key skills needed to develop the system
  • Software developers
  • Designers
  • Information architects
  • Others as appropriate
Whole Team

• The customers write stories and prioritize them

• The other members task out the stories and estimate them

• Members with appropriate skills estimate and perform tasks

• Tasks support the stories
Planning Game 1

• This is a periodic task (every 2 weeks) in which the customer chooses the high value features for the next release or iteration.

• Based on cost estimates from the developers.

• Estimates are not a contract, so re-steering is required throughout the iteration.
Planning Game 2

• Customer writes stories
• Developers estimate stories
• Customer prioritizes stories
• Developers give the “velocity”
• Customer chooses stories up to velocity
Planning Game 3

• Developers/Customer discuss stories

• Developers divide stories up into tasks

• Individual developer with appropriate skills chooses a task and estimates it

• If sum of task times > velocity then back to planning, otherwise build & test
Build Phase I

• Tracker keeps track of everyone’s progress.

• If all tasks/stories can’t be completed on time some are dropped. Customer chooses.

• At end of each task, all tests pass. Customer verifies

• If the customer still isn’t happy, write a new story
Build Phase 2

• If developers finish early, go back to customer for more work. Customer chooses

• Developers give a new “mini velocity”

• Next iteration velocity is adjusted based on what we complete this iteration
Practices--Developer

• Standup Meeting
• Sustainable Pace
• Coding Standard
• Test Driven Development
• Collective Code Ownership
• Small Releases
Practices--Developer

• Pair Programming
• Constant Refactoring
• Continuous Integration
• Simple Design
• Metaphor
• Retrospectives
New Practices

• The above practices may not all be appropriate as stated for an integrated team

• Practices are built on principles to give benefits

• Need to discover and implement appropriate practices for THIS team to achieve desired goals