Perspectives on eXtreme Programming (XP)

Steven Fraser
sdfraser@acm.org
Santa Clara, California
Outline

• Introduction to XP
• Perspectives on XP
• Discussion
XP Beginnings?

- A set of 12 integrated software engineering and business best-practices
- The C3 project @ Chrysler (1996)
  - In Kent’s words a “physics experiment”
Kent Beck’s Simplification

In my mind as we began XP as a physics experiment, where you remove all the variables possible so what you're left with is repeatable. Some of the usual variables we eliminated:

- Geographic separation
- Multiple customers
- Expensive deployment
- Stupid programmers
- Growth-averse database technology
- Computer-oriented programming language
- GUI-intense system
- Impersonal (>15 person) team
- Wildly changing requirements (replacing a legacy system)
- Disinterested business sponsors
XP Constraints

• Business/Requirements Engagement
  – Focused customers (on-site)
  – Engaged business sponsors

• Effective Teams
  – < 15
  – Standards
  – Pair programmers
  – Co-located teams

• Increasing Refactorability
  – Agile databases
  – Understandable code

• Constraining Complexity
  – Non-GUI-intense systems
  – Simplified deployment strategies
XP “Values”

- Simplicity
- Communication
- Feedback
- Courage

Beck, xTreme Programming Explained (00)
XP “Control Variables”

- Cost
- Time
- Quality
- Scope

Beck, xTreme Programming Explained (00)
The 12 XP Practices

- Planning Game
- Small Releases
- Metaphor
- On-site Customer
- Simple Design
- Pair Programming
- Test-Driven Design
- Refactoring
- Continuous Integration
- Collective Ownership
- Coding Standards
- Sustainable Pace

http://www.extremeprogramming.org
XP Practices Elaborated

- Customers define application features with user "stories"
- A customer proxy is onsite throughout the project
- Automated unit tests developed and run continuously
- Teams put small code releases into production
- Teams use a common system of names and descriptions
- Simply written, understandable code meets requirements
- Code appears in a consistent style
- Teams frequently revise the overall code design
- Programmers work side-by-side in pairs
- Teams have collective ownership of the code
- Teams integrate code and release it to a repository every few hours
- Teams work at a sustainable pace (no extended overtime)
### XP Process Increment

<table>
<thead>
<tr>
<th>Activity</th>
<th>Time-Scale</th>
</tr>
</thead>
<tbody>
<tr>
<td>Increment</td>
<td>2-3 Months</td>
</tr>
<tr>
<td>Iteration</td>
<td>2-3 Weeks</td>
</tr>
<tr>
<td>Acceptance Testing</td>
<td>1-2 Days</td>
</tr>
<tr>
<td>Stand-up Meeting</td>
<td>Once a Day</td>
</tr>
<tr>
<td>Pair Partnering</td>
<td>Hours - Days</td>
</tr>
<tr>
<td>Unit Testing</td>
<td>Minutes - Hours</td>
</tr>
<tr>
<td>Pair Programming</td>
<td>Minutes - Hours</td>
</tr>
</tbody>
</table>

- **User Stories**
- **Release Planning**
- **Iteration**
- **Acceptance Testing**
- **Small Releases**
XP Coding Iteration

User Stories → Develop Unit Test → Re-Pair Pair Program Refactor → Testing and Continuous Integration
## Xtreme Software Engineering?

<table>
<thead>
<tr>
<th>S/W Best Practice</th>
<th>Integrated into XP?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>Iteration Planning/Incremental Release</td>
</tr>
<tr>
<td>Requirements Validation</td>
<td>On-site Customer/Stories</td>
</tr>
<tr>
<td>Design</td>
<td>Refactor</td>
</tr>
<tr>
<td>Iterations</td>
<td>Short-short Iterations</td>
</tr>
<tr>
<td>Regular Integration</td>
<td>Hourly/Daily Integration</td>
</tr>
<tr>
<td>Code Inspections</td>
<td>Pair Programming</td>
</tr>
<tr>
<td>Module Ownership &amp; Standards</td>
<td>Collective Ownership &amp; Standards</td>
</tr>
<tr>
<td>Testing</td>
<td>Automated Testing</td>
</tr>
<tr>
<td>Architecture</td>
<td>Metaphor (System of Names)</td>
</tr>
</tbody>
</table>
XP Perspectives

- Are all XP practices required?
- Will XP scale for systems/teams?
- How effective is pair programming?
- XP - “Hacker and Hero”?
- On-site customer engagement?
- XP project initiation?
Are all XP practices required?

• Some would argue yes …
• Some would argue no …

…it depends!
Perspectives on XP Practices

• It isn’t XP if all 12 practices aren’t used

• Additional practices are required, e.g. napping  [Beck @ OOPSLA’02]

• Some practices are optional

• Do what is “right”
Will XP scale for systems/teams?

• It doesn’t matter since if you need a big team - XP is not the discipline of choice

• Scale by introducing teams of teams
How effective is pair programming?

• Evidence indicates – yes  [Williams & Kessler IEEE Software’00]

• It depends…
XP - “Hacker and Hero”? 

- Skeptics argue that XP is an opportunity for a reversal of roles between zoo-keepers and their charges
- Converts suggest that XP success speaks for itself
On-site customer engagement?

- Budget for the customer
- Motivate the customer
How to initiate an XP project?

- Executive overview to engage sponsors
- Team XP overview tutorial
- Team XP practices/tools/standards tutorial
- Facilitate a “planning game”
- Coaching services (pair programming, test-first, …)
- Evaluate successes and challenges – iterate!
XP Technology Transfer Questions

• Is the technology ready for “prime-time”?  
• Is there a business case?  
• Are the sponsors engaged?  
• What is the critical path?
Mechanisms for Tech Transfer

• Team tutorials, seminars, case studies
• Share demos and success stories
• Engage consultants, project managers, sponsors
• Establish pilot projects, news groups, forums…
• Manage logistics and schedule
• Assess impact and plan for follow-up
"Tech Transfer" Success Measures

<table>
<thead>
<tr>
<th>Category</th>
<th>Success Measures</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revenue</td>
<td>$, Customer Satisfaction, Success Stories</td>
</tr>
<tr>
<td>Expense</td>
<td>Time-to-Market, Tools, Consultants, Staff, Education, Infrastructure</td>
</tr>
<tr>
<td>Quality</td>
<td>Conformance Audits &amp; Metrics</td>
</tr>
<tr>
<td>Community</td>
<td>Size, Stability &amp; Growth</td>
</tr>
</tbody>
</table>

Success Momentum and Critical Mass
XP - Nirvana or Nemesis?

• Nirvana?
  – Cool Technology
  – Cost-Effective
  – Rapid Delivery
  – Customer Satisfaction

• Nemesis?
  – Real-time Systems
  – System/Team Size
  – Legacy Dependencies
  – Distributed Teams
  – Customer Interaction
  – Resistance to Change
Summary

eXtreme Programming
  – Bundles best-practices
  – Is a team activity
  – Requires discipline
  – Continues to evolve
For more Information:

www.xprogramming.org
http://www.extremeprogramming.org/
http://www.xp2003.org
http://www.nalusda.gov/ttic/test1.htm
sdfraser@acm.org


About the Speaker

Fraser is a California based consultant specializing in technology transfer, program management, and software process engineering. He spent more than 10 years with Nortel Networks in a variety of roles including: Process Architect, Senior Manager, and Disruptive Technologist.

From 1992 to 2001, Fraser was the Chair and Event Director of the Nortel Networks Design Forum, a proprietary global technology transfer event facilitated by interactive video conferencing, audio conferencing, web casting, and face-to-face interaction.

In 1994, he was a Visiting Scientist at the Software Engineering Institute (SEI) collaborating with the Application of Software Models project on the development of team-based domain analysis techniques. He completed his Doctoral studies in Electrical Engineering at McGill University in Montreal, Quebec, Canada.