Extreme Programming (XP): Strengths and Weaknesses

Amund Tveit
Department of Computer and Information Science
Norwegian University of Science and Technology

amund.tveit@idi.ntnu.no
http://www.idi.ntnu.no/~amundt/
+474 162-6572
What is Extreme Programming?

- **Software**
  - Most complex human artifact?

- **XP is a Software Engineering Methodology**
  - Lightweight * Agile Method

- **XP provides a set of:**
  - values
  - principles
  - practices

  ➔ Rapid dev. of high quality software
The Agile Manifesto

• Individuals and interaction
  – Vs. Processes and tools
• Working software
  – Vs. Large amounts of documentation
• Customer interaction
  – Vs. Contract negotiation
• React to changes (flexibility)
  – Vs. Follow pre-specified plan
Extreme Programming History

• Early 1990s – Kent Beck
• DaimlerChrysler 1996
  – C3 Project
• 2000 → now
  – Taking off
The 4 values of XP

1. Communication
   - Face-to-face
2. Simplicity
   - Optimize
3. Feedback
   - Users
4. Courage
   - This is not working
The 12 Practices of XP

1. Planning Game
2. Small Releases
3. System Metaphor
4. Simple Design
5. Continuous Testing
6. Refactoring
7. Pair Programming
8. Coll. Code Ownership
9. Continuous Integration
10. 40-Hour Work Week
11. On-Site Customer
12. Coding Standards
Practice 1. Planning Game

• User Stories
• Story Estimation
• Story Selection
Practice 2. Small Releases

• Useful features
• Incremental feature adding
Practice 3. System Metaphor

• Common Vision
• Naming convention
Practice 4. Simple Design

• Meet today’s requirements
• Simplest design doing the job
Practice 5. Continuous Testing

- Unit Tests
- Acceptance Tests
Practice 6. Refactoring

• Duplicate Code
• Improve Code
• Restructure
• Maintainability
• Readability
• Retest
Practice 7. Pair Programming

• Code
• Test
• Shift/Rotate
Practice 8. Collective Code Ownership

- Multiowned code
- Location indep.
Practice 9. Continuous Integration

• Daily build
• Pre-test
• Post-test
Practice 10. 40-Hour Work Week

- Home on time
- Crunch overtime
- Bad symptom
Practice 11. On-site Customer

• Real User
• Customer Proxy
• Product manager
Practice 12. Coding Standards

• "Unfingerprinted"
• Readable
Two Important XP Acronyms

• **YAGNI** = You aren’t gonna need it
  ➔ don’t add nice-to-have features

• **DTSTTCPW** = Do the simplest thing that could possibly work
  ➔ don’t add unjustified complexity
Typical XP Project

1. Programmers in same room
2. Fixed Iteration Cycles
3. Write Test first
4. Working system at end of cycles
5. Virtually Defect-free
6. Scale ➔ dozen (12) programmers
Extreme Programming Project
XPs Claimed Cost of Change

Figure 1. Relative cost of change: Traditional versus XP.
Little documentation using XP

Strengths
• Saving total time
• Less non-code maintenance

Weaknesses
• Customer dependency on initial developers
• Potential distortion of competition
• Hard to outsource?
The Planning Game

Strengths

• Reduce likelihood of unexpected complications
• Team shares understanding
• Sharing aids finding weak spots

Weaknesses

• Quick design sessions are too quick (10 min)
• Can lead to bad design decisions
Unit Testing

Strengths
- Explains code with test cases
- Unit tests are up-to-date documentation
- Hypothesis for new programmer ➔ unit tests

Weaknesses
- Incentive for not writing documentation
- Unit tests for databases, distributed, realtime and GUI hard to test with unit tests
Refactoring

**Strengths**
- Give programmers the right to improve their code
- Refactoring makes the programmer understand the system better
- Readability

**Weaknesses**
- Incentive to refactor instead of commenting (Fowler)
- Software improvement is subjective, can confuse others
Simple design

**Strengths**
- Fights over-designs
- Maximal simplicity eases program understanding

**Weaknesses**
- Hard to create good corresponding metaphors
- Simplistic design choices can constrain future changes
## Collective Code Ownership

<table>
<thead>
<tr>
<th>Strengths</th>
<th>Weaknesses</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Less dependency on individuals (risk)</td>
<td>• Assumes others take responsibility for code (single ownership nurtures responsibility?)</td>
</tr>
</tbody>
</table>
Pair Programming

Strengths

• Larger ”solution space” with pair
• Emergence of ideas (1+1 > 2)
• Fewer defects
• Readability
• Co-Learning
• Code co-ownership

Weaknesses

• Avoidance of explicit code reviews
• Increased initial costs (15%)
Individuals in Pairs

Strengths
- Less goofing off
- More careful coding

Weaknesses
- Constant interaction is tiring
- Lot of noise produced
- Ergonomic problems
- Groupthink can occur (1+1 < 2)
- Less rest
The Customer

Strengths

• Emphasis on customer involvement
• Programmer est. before committing to plans
• Emphasis on responsibility for quality

Weaknesses

• Senior customer hard to get ~ 24x7
• Customer may dislike the "variable scope" of XP
  ➔ House building analogy
• We didn’t have time to build an entrance door!
Other XP strengths and weaknesses

**Strengths**

- Continuous reviews to improve team performance
- Having engineers manage functional content
- System running at all times

**Weaknesses**

- Probably not suitable for large-scale projects
- Probably not suitable for mission critical systems (e.g. Military, Nuclear)
  ➔ need formal validation
Other Agile Methods

- Scrum
- Dynamic Systems Development Method
- Crystal Family
- Feature Driven Development
- Adaptive Software Development
How can XP weaknesses be handled?

• Designing up-front
  – vs. Architectural spike
• Flexible pair programming
• Handling dependency between stories
• Forward refactoring for reuse
  – Versus improvement
• Integrate Organizatorical Aspects
Personal XP-related Experiences

• Unit Testing - jfipa

• Pair Programming
Conclusion - strengths and weaknesses

**Strengths**

- Trusting the developer
- Flexibility
- Focus on code (the product) instead of model

**Weaknesses**

- Scalability
- Customer reluctancy
- Project leader scepticism
Presentation Acknowledgements

**Primarily:** Hans Gallis (Simula Research, Oslo)

- Dr. Stephen Willmott, (Catalunya Univ., Spain)
- Dr. Reidar Conradi, IDI/NTNU
- Dr. Torgeir Dingsøyr, Sintef, Trondheim
- Dr. Alf Inge Wang, IDI/NTNU
- Rolv Inge Seehus, IDI/NTNU
- Carl-Fredrik Sørensen, IDI/NTNU
- Thomas Brox Røst, IDI/NTNU