12. Extreme Programming

Objective

- To minimize the risk in development
  - Understand requirements better
  - Be ready to change as requirements change
- To succeed in the development process
- To complete the project
  - in budget
  - on time
- If the project has to be cancelled, do so with minimal damage
- Create a system that is
  - easier to maintain
  - less expensive to evolve
- Keep the bug count low
What about extensibility?

- Your system should be able to change with least cost
- You should anticipate change?
- Does it mean that you build for what you think may be needed?
- It depends
- Here are questions to ask

Cost of the new feature

- What are the chances you will need to add new feature?
- How much does it take now to provide it?
- What is the worth of that feature to customer?
- How much will it cost to provide it in the future?

- If it will cost almost the same in the future, and you are not certain of the feature’s worth, it may be better to wait
  - If the features are important, we can implement it later
  - If it is not needed, we did not implement it
So Should I not worry about extensibility?

• You should!
• However, there are ways to address it
• Check on your ability to anticipate the need and change
• Check on your ability to build the system so the change in the future is incremental
• Refactor the system as it evolved

Control Variables

• Cost
  – Too little, does not solve problems
  – Too much, some times more of a problem
• Time
  – More time can improve quality and increase scope.
  – Too much time hurts as well
    • Feedback from system in production is imperative
• Quality
  – Sacrificing this may result in short term gains
  – Over the long haul, lost is enormous
• Scope
  – Lesser the scope, better the quality
  – You can deliver sooner as well
  – Assuming it meets the business needs
Set of Values

- Communication
  - Need to communicate critical change in requirements, design, etc.
  - Put in place practices that will enhance communication
- Simplicity
  - Find simplest thing that will work
  - Build some thing simple today and pay a little to change tomorrow than build some thing complicated today that may never be used
- Feedback
  - Unit tests provide feedback
  - Corrected in minutes and days, not weeks
  - A system that stays out of the hands of users is trouble waiting to happen
- Courage
  - Do not hesitate to throw code away if you find a better simpler way
  - Do not hesitate to call attention to problems if they are significant and will benefit from reworking

What’s XP?

- Kent Beck
- XP has nothing new, yet it has something new!
- It takes good commonsense principles and practices to extreme levels
  - If code review is good, we’ll review code all the time
    - Pair programming
  - If testing is good, every body will test all the time
    - Unit testing by developers, functional testing by customers
  - If design is good, we’ll make it part of everybody’s daily business
    - Refactoring
  - If simplicity is good, we’ll make it part of the system with simplest design that supports its current functionality
  - If architecture is important, everybody will work defining and refining the architecture all the time
    - metaphor
  - If integration testing is important, then we’ll integrate and test several times a day
    - Continuous integration
  - If short iterations are good, we’ll make the iterations really, really short – seconds and minutes and minutes and hours, not weeks and months and years
    - The planning game
XP Principles

- The Planning Game
  - Scope next release with business priorities, technical estimates
  - Update the plan based on reality
- Small Releases
  - Put simply system into production quickly
  - Release new version in short cycle
- Metaphor
  - Guide development with simple shared story of how the whole system works
- Simple Design
  - Design as simple as possible at any given moment.
- Testing
  - Continually write and run unit tests

XP Principles

- Refactoring
  - Restructure system without changing its behavior to remove duplication, improve communication, add flexibility and simplify
- Pair Programming
  - Two programmers, one machine, four eyes are better than two
- Collective Ownership
  - Anyone can change code anywhere in the system at any time
- Continuous Integration
  - Integrate and build the system many times a day, every time a talk is completed.
- 40-hour Week
  - Never work overtime a second week in a row
- On-site Customer
  - Real, live user on the team, available full-time to answer questions
- Coding Standards
  - All code written accordance with rules emphasizing communication through the code
Where does XP work?

• Culture
  – Business culture
    • How change is accepted? Need to work long hours? Goal oriented? Heavy on paper work?

• Size
  – Team size of around 10 is ideal

• Technology
  – Must be able to make change quickly and get feedback

• Work environment
  – Should promote closer interaction and communication