Extreme Programming (XP)

Agile Software Development Paradigm

- Values individuals and interactions over processes and tools.
- Values working software over comprehensive documentation.
- Values customer collaboration over contract negotiation.
- Values responding to change over following a plan.

Agile Methodologies

- Extreme Programming
- Adaptive Software Development
- Crystal
- Dynamic Systems Development Method
- Feature-Driven Development
- Pragmatic Programming
- Scrum

The Cost of Change

- Traditional methodologies assume the cost of change rises rapidly (exponentially?) over the life cycle of software.
- XP assumes that under the right conditions the cost of change can be held constant.
- These assumptions have a dramatic impact on the development process.
- There is little empirical evidence to support either assumption. Studies are either too old or too small.

Development as Maintenance

- In an XP project there is a working system almost from the very beginning.
- XP development is *extremely* incremental. Requirements, design, and implementation are all developed in very small increments.
- Each time a new “story” is elicited from the customer, it is implemented as quickly as possible and delivered to the customer for feedback.

Roles of Developers

- In XP, each development team member works on all phases of development: analysis, design, implementation and testing.
  - This is in contrast to traditional methods where people are specialists, and workers further down stream are considered less skilled and are lower paid.
- In XP, it is assumed that incremental development will continue over the entire life of the software.
- There is no distinction between developers and maintainers.

Documentation

- XP projects do not produce nearly as much documentation as traditional approaches.
- The code is expected to express its intent without the need for additional documentation.
- There is less need for documentation, since the same team members are involved in all phases of development. The project is not passed off from one group of specialists to another.
- It is assumed that although there may be some turn over of personnel, the same team will be responsible for the project over its entire lifecycle.

XP Practices (1)

- The Planning Game
  - Quickly determine the scope of the next release by combining business priorities and technical estimates.
- Small Releases
  - Put a simple system into production quickly, then release new versions on a very short cycle (one or two months).
- Metaphor
  - Guide all development with a simple shared story of how the whole system works.
  - Replaces most system architecture.

XP Practices (2)

- Simple Design
  - The system design is kept as simple as possible at all times. The design is only as complex as necessary to support the current development cycle.
  - There is no designing for the future.
  - Extra complexity is removed whenever it is discovered.
  - The design is “in the code”.
- Testing
  - Programmers write unit tests before writing implementation code. All unit tests must pass for development to continue.
  - Customers write functional tests based on stories.

XP Practices (3)

- Refactoring
  - Programmers restructure the system without changing its behavior to remove duplication, improve communication, simplify, or add flexibility.
- Pair programming
  - All production code is written with two programmers at one machine.
- Collective ownership
  - Anyone can change any code anywhere in the system at anytime.

XP Practices (4)

- Continuous integration
  - Integrate and build the system many times a day, every time a task is completed. All unit tests must pass after every build, or the change is withdrawn.
- 40-hour week
  - Never work overtime in two consecutive weeks.
- On-site customer
  - Include a real, live user on the team, available full-time to answer questions.
- Coding standards
  - Standard must be adopted voluntarily by all team members.

XP Examined

- XP was designed to be a lightweight process for small projects with 2 - 10 developers.
- It is probably not suitable for larger projects.
- Many individual XP practices would be very bad ideas in traditional approaches, but might work when used together.
- Other XP practices might be useful outside of XP:
  - e.g. testing, pair programming, 40-hour week