Introduction to Extreme Programming

John Brewer • Jera Design • jbrewer@jera.com

http://www.jera.com/techinfo/

The 12 Core Practices of XP:

1. **The Planning Game**: Business and development cooperate to produce the maximum business value as rapidly as possible. The planning game happens at various scales, but the basic rules are always the same:
   a. Business comes up with a list of desired features for the system. Each feature is written out as a **User Story**, which gives the feature a name, and describes in broad strokes what is required. User stories are typically written on 4x6 cards.
   b. Development estimates how much effort each story will take, and how much effort the team can produce in a given time interval (the iteration).
   c. Business then decides which stories to implement in what order, as well as when and how often to produce a production releases of the system.
2. **Small Releases**: Start with the smallest useful feature set. Release early and often, adding a few features each time.
3. **System Metaphor**: Each project has an organizing metaphor, which provides an easy to remember naming convention.
4. **Simple Design**: Always use the simplest possible design that gets the job done. The requirements will change tomorrow, so only do what's needed to meet today's requirements.
5. **Continuous Testing**: Before programmers add a feature, they write a test for it. When the suite runs, the job is done. Tests in XP come in two basic flavors.
   a. **Unit Tests** are automated tests written by the developers to test functionality as they write it. Each unit test typically tests only a single class, or a small cluster of classes. Unit tests are typically written using a unit testing framework, such as JUnit.
   b. **Acceptance Tests** (also known as **Functional Tests**) are specified by the customer to test that the overall system is functioning as specified. Acceptance tests typically test the entire system, or some large chunk of it. When all the acceptance tests pass for a given user story, that story is considered complete. At the very least, an acceptance test could consist of a script of user interface actions and expected results that a human can run. Ideally acceptance tests should be automated, either using the unit testing framework, or a separate acceptance testing framework.
6. **Refactoring**: Programmers refactor code as needed to keep the code as simple as possible. (See right column for XP's definition of simple.) You can do this with confidence that you didn't break anything because you have the tests.
7. **Pair Programming**: All production code is written by two programmers sitting at one machine. Essentially, all code is reviewed as it is written. (You can still write non-production experimental code solo.)
8. **Collective Code Ownership**: No single person "owns" a module. Any developer is expected to be able to work on any part of the codebase at any time.
9. **Continuous Integration**: All changes are integrated into the codebase at least daily. The tests have to run 100% both before and after integration.
10. **40-Hour Work Week**: Programmers go home on time. In crunch mode, up to one week of overtime is allowed. But multiple consecutive weeks of overtime are treated as a sign that something is very wrong with the process.
11. **On-site Customer**: Development team has continuous access to a real live customer, that is, someone who will actually be using the system. For commercial software with lots of customers, a customer proxy (usually the product manager) is used instead.
12. **Coding Standards**: Everyone codes to the same standards. Ideally, you shouldn't be able to tell by looking at it who on the team has touched a specific piece of code.

What does "Simplest" mean?

XP actually has a very specific definition of "simplest" (based on the list in Extreme Programming Explained, p.109):

1. The system (code plus tests) clearly communicates everything that needs to be communicated at the current instant in its development. This means that it runs every existing test, and that the source code clearly reveals the intention behind it to anyone who reads it.
2. The system contains no duplicate code, unless that would violate (1).
3. The system contains the minimum number of classes possible without violating (1) or (2).
4. The system contains the minimum number of methods possible, consistent with (1) (2) and (3).

Some Common XP Terms

Business the part of an organization that wants a program written, usually because they can make or save money by using it themselves, or make money by selling it.

Customer a person or group of people who represent the interests of business to the development team. The ideal customer is either a user of the system, or a proxy for the users, such as a product manager.

Development the part of an organization that writes programs, usually to meet the needs/requirements of business.

Iteration a period of fixed duration (typically 1, 2 or 3 weeks) during which a set of features is added to the system.

Story a description, written by the customer, of a single desired additional feature of the system being developed. Typically written on a 4x6 card.

XP Resources

Books

Extreme Programming Explained by Kent Beck. Addison-Wesley 2000. The first book on XP published, and still the best introduction.

Refactoring: Improving the Design of Existing Code by Martin Fowler. Addison-Wesley 1999. The definitive book on refactoring. Refactoring is essential to XP, but still extremely useful to anyone engaged in object-oriented programming.

Extreme Programming Installed by Ron Jeffries, Ann Anderson, and Chet Hendrickson. Addison-Wesley 2001. A book on XP by and for programmers. Lots of good real-world advice from people who have been doing XP the longest.

Planning Extreme Programming by Kent Beck and Martin Fowler. Addison-Wesley 2001. An entire book on the XP planning practice, going over iteration and release planning in detail.

Web Sites

Extreme Programming FAQ maintained by John Brewer. http://www.jera.com/techinfo/xpfaq.html

C2 Wiki a user-editable and extensible site containing lots of information about XP and patterns. http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap

Software

JUnit a unit testing framework for Java. http://www.junit.org/

xUnit Unit testing frameworks for other languages. http://www.xprogramming.com/software.htm

Groups

extremeprogramming public Yahoo! Groups mailing list on Extreme Programming. http://groups.yahoo.com/group/extremeprogramming

BayXP Bay Area XP Users Group http://www.jera.com/bayxp/