MANIFESTO FOR AGILE SOFTWARE DEVELOPMENT

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

INDIVIDUALS AND INTERACTIONS over processes and tools
WORKING SOFTWARE over comprehensive documentation
CUSTOMER COLLABORATION over contract negotiation
RESPONDING TO CHANGE over following a plan

That is, while there is value in the items on the right,
we value the items on the left more.

Kent Beck
Mike Beedle
Arie van Bennekum
Alistair Cockburn
Ward Cunningham
Martin Fowler
James Grenning
Jim Highsmith
Andrew Hunt
Ron Jeffries
Jon Kern
Brian Marick
Robert C. Martin
Steve Mellor
Ken Schwaber
Jeff Sutherland
Dave Thomas

AGILE SOFTWARE DEVELOPMENT CONSORTIUM

Representatives from

Extreme Programming
http://www.extremeprogramming.org/

SCRUM
http://www.controlchaos.com/

DSDM
Dynamic **S**ystems **D**evelopment **M**ethod
http://www.dsdm.org/

Adaptive Software Development
http://www.dorsethouse.com/books/asd.html

Crystal
http://crystalmethodologies.org/

Feature-Driven Development
http://www.featuredrivendevelopment.com/

Pragmatic Programming
http://www.pragmaticprogrammer.com/

PRINCIPLES BEHIND THE AGILE MANIFESTO

We follow these principles:

- Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

- Business people and developers must work together daily throughout the project.

- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

- Working software is the primary measure of progress.

- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

- Continuous attention to technical excellence and good design enhances agility.

- Simplicity--the art of maximizing the amount of work not done--is essential.

- The best architectures, requirements, and designs emerge from self-organizing teams.

- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

http://www.agilealliance.com/home

XP's 12 CORE PRACTICES

1

Customers define application features with user stories.

2

XP teams put small code releases into production early.

3

XP teams use a common system of names and descriptions.

4

Teams emphasize simply written, object-oriented code that meets requirements.

5

Designers write automated unit tests upfront and run them throughout the project.

6

XP teams frequently revise and edit the overall code design, a process called refactoring.

7

Programmers work side by side in pairs, continually seeing and discussing each other's code.

8

All programmers have collective ownership of the code and the ability to change it.

9

XP teams integrate code and release it to a repository every few hours and in no case hold on to it longer than a day.

10

Programmers work only 40 hours per week; there's no overtime.

11

A customer representative remains on-site throughout the development project.

12

Programmers must follow a common coding standard so all the code in the system looks as if it was written by a single individual.

What is **Pair Programming**?

TWO programmers working side-by-side, collaborating on the same design, algorithm, code or test. One programmer, the driver, has control of the keyboard/mouse and actively implements the program. The other programmer, the observer, continuously observes the work of the driver to identify tactical (syntactic, spelling, etc.) defects and also thinks strategically about the direction of the work. On demand, the two programmers can brainstorm any challenging problem. Because the two programmers periodically switch roles, they work together as equals to develop software.

-- Laurie Williams
North Carolina State University Computer Science

CONCLUSION

"All methodologies are based on fear. You try to set up habits that prevent your fears from becoming reality."

KENT BECK

"XP reflects my fears:

- Doing work that doesn't matter
- Having projects canceled because I didn't make enough technical progress
- Making business decisions badly
- Having business people make technical decisions badly for me
- Coming to the end of a career of building systems and realizing that I should have spent more time with my kids
- Doing work I'm not proud of"

“XP also reflects things I’m not afraid of:

- Coding
- Changing my mind
- Proceeding without knowing everything about the future
- Relying on other people
- Changing the analysis and design of a running system
- Writing tests

I had to learn not to fear these things.”

"…we felt privileged to work with a group of people who held a set of compatible values, a set of values based on trust and respect for each other and promoting organizational models based on people, collaboration, and building the types of organizational communities in which we would want to work. At the core, I believe Agile Methodologists are really about "mushy" stuff – about delivering good products …about the mushy stuff of values and culture."

Jim Highsmith
For the Agile Alliance
http://www.agilemanifesto.org/history.html

More Quotes:

"XP has traditionally worked best in small or medium-size teams composed of competent developers who work well together; where the customer's requirements may change frequently; and where frequent small releases are possible."

"Five Lessons You Should Learn from Extreme Programming"

1. Code for Maintainability
2. Know your status
3. Communicate early and often
4. Do things that matter
5. Fix your most important problem first

http://www.onlamp.com/lpt/a/4061

One more Quote:

"One thing that's good about XP is that it simplifies things developers don't classically like to do, like testing and code review. And anything that makes developers do that is a desirable thing. But right now, there isn't enough evidence that XP is a breakthrough that all teams should embrace."

http://www.computerworld.com/printthis/2001/0,4814,66192,00.html
Computerworld, 2001

Another Quote:

"…XP requires unfailing discipline from every member of the team throughout the project. This makes it anything but lightweight. Additionally, the 12 practices are sot tightly dependent on each other that tailoring XP (or skipping a few of the practices) can be tricky."

"The Irony of Extreme Programming"
Matt Stephens and Doug Rosenberg
Dr. Dobb's Journal, May 2004

One more quote:

"We can drive ourselves crazy with expectation. But by preparing for every eventuality we can think of, we leave ourselves vulnerable to the eventualities we can't imagine."

eXtreme Programming explained: Embrace Change, Kent Beck, Addison-Wesley, 2000.