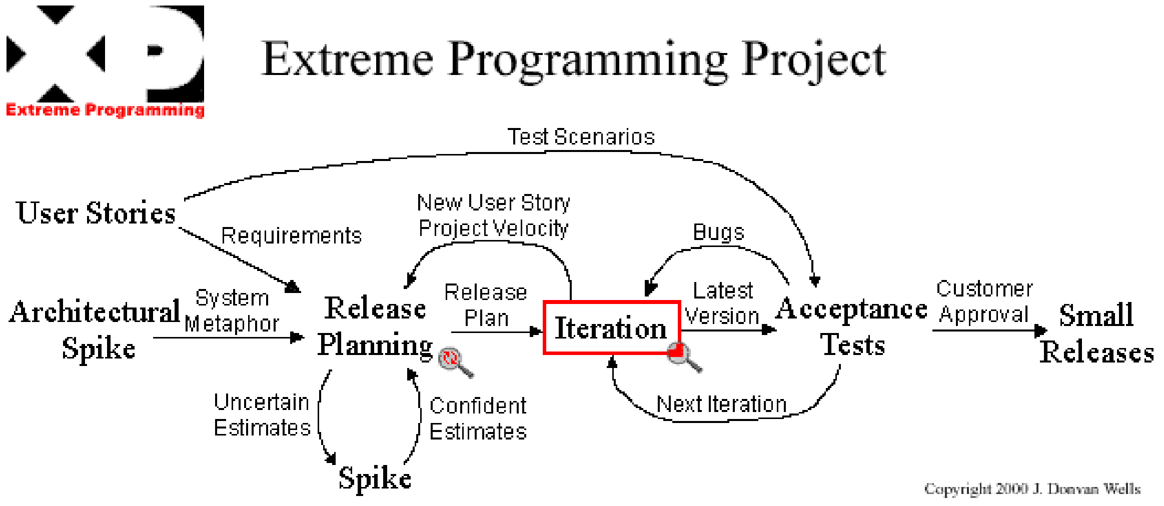
XTREME PROGRAMMING IN CF

By Michael Smith, TeraTech, Inc Michael@teratech.com



The Four Project Values

Simplicity + Communication + Testing = Aggressiveness

Simplicity

We strive always to "do the simplest thing that could possibly work". We're quite serious about this. We have found that with the right architecture and correctly-factored code, we can rapidly extend our software when the need arises. If we work on something "we're gonna need", we're not working on something we DO need.

Communication

Good communication in every way. In our code, we all use the same formatting conventions, naming conventions, and coding standards.

Testing

We are fanatics for testing. We would like to have more lines of test than we do of actual code. Every fuseaction must have a corresponding unit test page. There are 1000s of tests. When we release code, all the unit tests must run at 100%. You can't release unless they do: if they don't, you fix the problem immediately.

We have functional tests as well, run by people testing the system functional stories. There are hundreds of functional tests.

The result of all this testing is that we know the system works, and when we make errors we find them immediately.

Aggressiveness (Fearlessness)
The result of the other three values is that we can be aggressive, or fearless. We can change any part of the system to be better because we know we have a solid system of tests. We can try things and if we don't like how they work, we throw them away and try again. We know we won't break the system, which gives us the confidence to move forward rapidly.

YANGNI You're NOT gonna need it!
Do the simplest thing that could possibly work

The Rules and Practices of Extreme Programming.

Planning
1. User stories are written.
2. Release planning creates the schedule.
3. Make frequent small releases.
4. The Project Velocity is measured.
5. The project is divided into iterations.
6. Iteration planning starts each iteration.
7. Move people around.
8. A stand-up meeting starts each day.
9. Fix XP when it breaks.

Designing
1. Simplicity.
2. Choose a system metaphor.
3. Use CRC cards for design sessions.
4. Create spike solutions to reduce risk.
5. No functionality is added early.
6. Refactor whenever and wherever possible.

Coding
1. The customer is always available.
2. Code must be written to agreed standards.
3. Code the unit test first.
4. All production code is pair programmed.
5. Only one pair integrates code at a time.
6. Integrate often.
7. Use collective code ownership.
8. Leave optimization till last.
9. No overtime.

Testing
1. All code must have unit tests.
2. All code must pass all unit tests before it can be released.
3. When a bug is found tests are created.
4. Acceptance tests are run often and the score is published.

Resources

http://www.extremeprogramming.org/
http://www.xprogramming.com/
http://www.c2.com/cgi/wiki?ExtremeProgrammingRoadmap

NOTES

This page is for your personal notes. Enjoy.