Lean, Agile, and Extreme Programming

By Mark Windholtz, ObjectWind Software Ltd,
(while working in the RailsStudio.com)

The following is an introduction and update to the 2nd edition of Extreme Programming process in the context of Lean Software Development and Agile the family of processes.

Introduction

Background

The de facto standard Industry practice in most Software development teams is either chaotic or bureaucratic. The chaotic style chief characteristic is "Code and Fix"; while the bureaucratic style's chief characteristic is waiting for someone else to complete a prerequisite tasks. Some of the indicators of either of these approaches are: a long integration phase, a long test phase, high defect counts.

A common solution of the 1980's and 1990's was to advocate a more rigorous and heavy process with lots of documents, and lots of control points. The metaphor for this style is that it is wise to have *Blue Prints before you build*.

The heavy weight process solution has been around for almost 20 years so why is it not already a common and often used practice? As it turns out, it has not been very successful. Lots of excuses are given. Mostly the people are blamed. As in: *if only our people would follow this great process!*

The truth is that heavy weight processes often set up conflicting goals.
People executing the process too often have to decide to support either the Process or get the work done. This is the conflict: Process vs. Product.
The heavy processes have also been criticized as bureaucratic & complicated. What's harder: building the s/w or implementing the entire RUP toolset and dashboard?
Big Processes often introduce added organizational conflicts, which distracts from the delivery of the product.

What is Agile?

Agile and Extreme Programming arose in the late 1990's as a compromise between no control (chaos) and no movement (bureaucracy). The focus is on code delivery, but the approach is highly disciplined and professional.

The figure below introduces the Agile processes landscape.

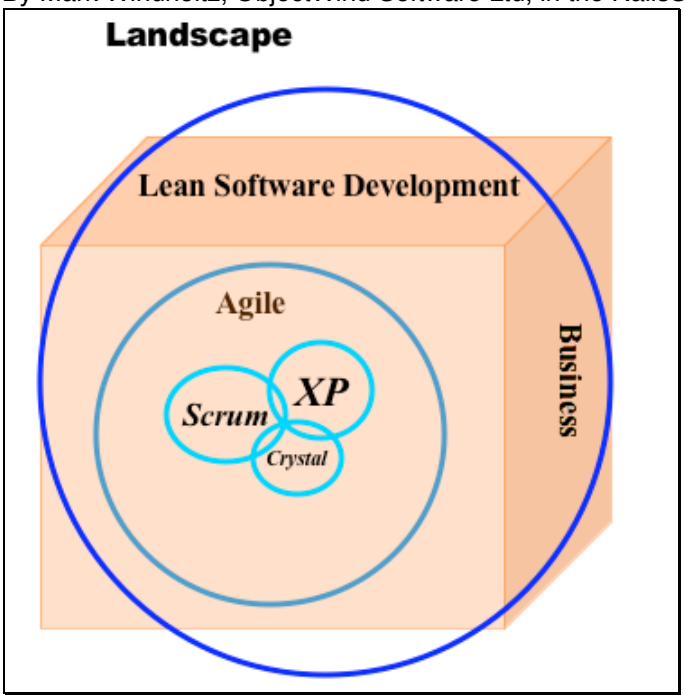


Figure 1 - Agile Landscape

Agile is the name for a family of processes, which include Extreme Programming (XP), Scrum, and Crystal. Each process has it's own guidelines for how to structure and complete programming work. They share a set of values expressed in figure 1.

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.

Figure 2 - Agile Values

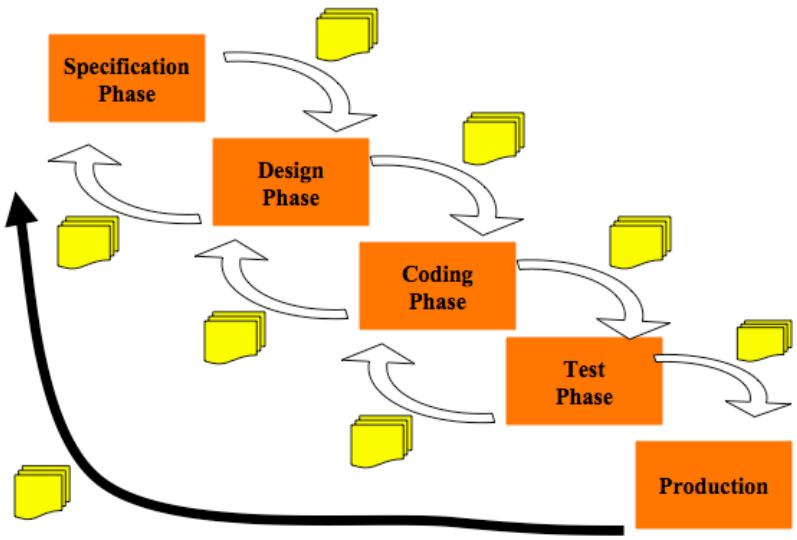
By Mark Windholtz, ObjectWind Software Ltd, in the RailsStudio.com

Lean Software Development

The Agile family of processes exists inside a larger approach to software development that is named ***Lean Software Development.*** The Lean approach brings the business model (implemented by Toyota, Wal-mart, South West Airlines, and others) and applies it to software development. Some fundamental tenants of Lean-Thinking are the elimination of waste and the focus on real customer value.

A key observation of Lean-Thinking is that Inventory is a form of Waste.
Inventory is defined as any non-producing Asset. Inventory is any artifact not producing current income. Inventory reduces responsiveness.
In terms of software, application features currently being developed are Inventory. *Inventory increases risk* since we don't know if or how it will work until it actually does work. Inventory incurs high administrative costs because it requires storage and maintenance. It reduces response to changing markets

Phased software processes require more Inventory in the form of no-value artifacts between phases. Remember, true customer value is only in the finished, functioning software not in the intermediate paperwork generated. This is the key to understanding the business model and the Agile implementation of the business model.



The 7 Wastes of Software development

1. Inventory – Features not yet in production, intermediate documents
2. Defects not caught
3. Over -production, elaboration – code written "just in case" we may need it (but don't).
4. Extra Process Steps
5. Motion getting the needed information – walking between cubes on separate floors
6. Waiting for information
7. Transportation – handoffs, sign-offs, approvals

Just In Time Software

One company using Agile reported delivering 3 very complex web portals into production. The team deploys into production every 2 weeks. They encountered only one production defect in last 9 months

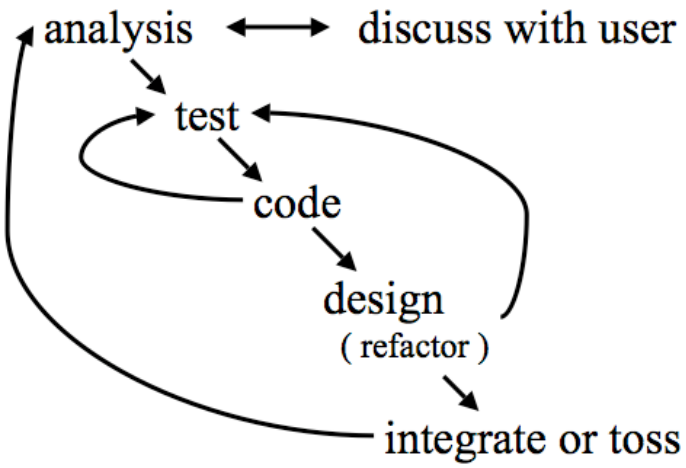
Extreme Programming

> XP is about delivering to deadline.
> Every two weeks you release something
> and every two weeks you check your plan.
> You get really good at getting stuff done.
>
> - James Grenning, ObjectMentor

Extreme Programming (XP) is a set of values, principles, and disciplined practices that focus the entire team on delivery of value and reduction of waste.
A typical day for the programming team would be as follows:

- 15 minute team *stand-up* meeting
- Get a story from task board
- Pair-up
- Talk with the customer sitting with the team
- Hold programming episodes
- Monitor the automated build & test system
- Deliver a few fully tested, integrated business Stories
- Go home at a reasonable time

The XP Episode Cycle starts with talking to the user about the task at hand. Then the two programmers, working together write an automated test for that task (about 10 minutes). The test should fail at this point. They then collaborate on writing the fewest lines of code to make the test succeed(about 15 minutes). Once the test passes they clean up the code and add better design structure to it (about 15 minutes). Finally the programming pair check-in their code into source code control. If integration is difficult, they will often throw the code away and re-implement it rather than struggle indefinitely with a difficult integration.



Another Agile company (www.lifeware.ch) reports running 4000 tests run with every change to the code. New features go into production every evening. Only needed workflows implemented. They advertise a low cost of operation since they only pay one-day of programmer salaries, before the work in earning money for the company.

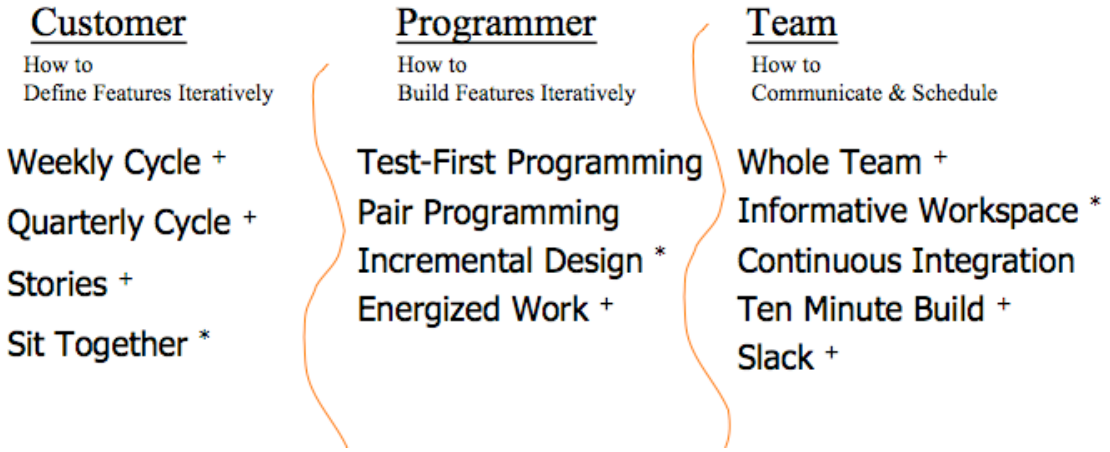
By Mark Windholtz, ObjectWind Software Ltd, in the RailsStudio.com

XP Values

Communication - let everyone know what's happening
Simplicity - don't solve problems you don't have
Feedback - start with the end in mind
Courage - effective Action in the face of fear
Respect - if people don't care, the project will not succeed

XP Primary Practices

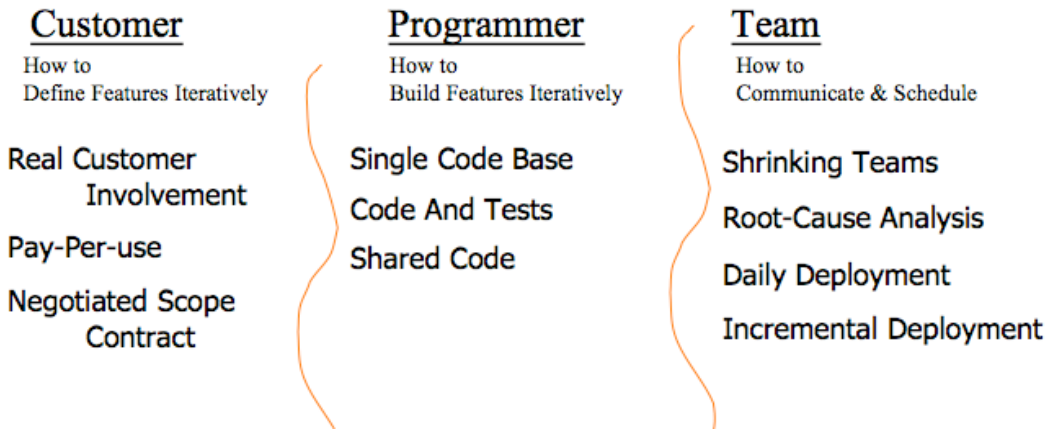
A new XP team should start by implementing as many of the following Primary Practice as possible. Take them one at a time, but do not stop until all have been added to your weekly process. It is often harder to do some of them than to do all of them together, since the practices are support each other. For instance, it is not wise to do Incremental Design unless Test-First Programming is widely used. Changing the design with a good test suite is easy; without a test suite it is dangerous. For further advice on implementing these practices see the books in the reference section. Or consult with one of your colleagues who have experience with the practice you are interested in.



XP Corollary Practices

The Corollary Practices are more advanced. And should be added only after the Primary practices are mostly in place.

By Mark Windholtz, ObjectWind Software Ltd, in the RailsStudio.com



Extreme Programming Explained: Embrace Change 2nd. Edition

by Kent Beck

Lean Software Development (An Agile Toolkit)

by Mary Poppendieck