Waste & Value

Part 1 of ObjectWind
Lean Software Development series *

Mark Windholtz
ObjectWind Software Ltd

* Lean Software Development, by Mary Poppendieck

ObjectWind Software

- **Lean Development**
  - Your team, or
  - Delivered applications
- **Training**
  - OO Design
  - Process Improvement
  - Unit Testing
  - Fitnesse Testing
  - Extreme Programming
- **Project Services**
  - Chartering
  - Reviews
  - Implementation
  - Lessons Learned
  - Extreme Programming

Dilbert


DOES ANYONE HAVE A SUGGESTION FOR REDUCING OUR INVENTORY?
LET'S SELL IT TO OUR CUSTOMERS.
www.dilbert.com scottadams@aol.com
Copyright © 2002 United Feature Syndicate, Inc.

Success Stories

- Wal-Mart
- Procter & Gamble
- South West Air
- Jet Blue
- Toyota
- Dell Computer

- Used **Lean-Thinking**
  - to transform their industry

Why do we build software?

- To make **Profit**!
- Profit == Revenue – Expense
- Revenue from Delivered Perceived **Value**
- Expense = Material + Labor + **Waste**

Value

- What is the Value?
- What makes money for the Customer
  - Lines of Code Written?
  - J2EE? .NET?
  - Reusable Book-keeping Framework?
  - First draft Design Document?
  - Well reviewed Design Document?
  - Ruby script written and in production in 10 hours to track baseball stats for talent scouts?

Real Value

- Not Technology
- Usable and actually Used
- Must be in production
- Focus on Usage
  - *Fitness for Use*
  - *Compliance to Specification*
- Who judges **Fitness for Use ?**

That's a waste of time ?

- Things I've heard:
  - In-process testing … is a waste of time.
  - Cleaning-up ugly code … iawot.
  - Finding abstractions … iawot.
  - Collaborative problem solving …. iawot.
  - Communication among programmers … iawot.

Waste

- Partially Finished Work
- Extra Features
- Extra Process Steps
- Task Switching
- Waiting
- Motion
- Defects
- Management Activities

Partially Finished Work

- A feature only makes money in production
- Risk > 0% that might not deliver
- Ties up resources
- Becomes obsolete (4% per month)
- Revenue – Expense of that feature

- To do: define complete, small tasks, that provide real value. And deliver.

Extra Features

- *Just in case* programming
- *Just in case* requirements
  - Standish Group Study: 45% feature not used
- Last chance before requirements are frozen, so stick everything in as high priority!
- More complexity creates more expense starting NOW
  - More code to be tracked and understood
  - Why is this here?
- To do: Focus on the current needs.
  - Build it clean and well-tested. Deliver.

Task Switching

- Multiple projects cause interrupts.
- Interrupts leave partially finished work
- Switch over time in not Zero
- Switch over time adds no value

- To do two project fast, do one at a time
- Do one task at a time to completion. Deliver.

Waiting

- Waiting for
  - project kick-off
  - Design review
  - Decision meetings
  - Answers from a person sick at home
- While waiting the real world changes
- Value is not delivered
- Missed opportunity to make Profits
- To do: Apply Queuing Theory.
  - Overlapping skills. Smaller deliveries.

Motion

- *I'm going upstairs to see the test results*
- Going to find information is waste
- Moving Documents between groups
  - Requirements sent to Designers
  - Designs sent to programmers
- Manual Testing
- Manual Build/Deploy Process
- To do: Automate everything
  - Build Information Radiators

Defects

... yawn ...

Amount of waste

= impact of defect * time undiscovered

- To do: Find defects as they are coded
  - Use Test First Development
- Strong positive correlation between number of auto-tests written and programmer productivity

Management Activities

- Value of Project tracking and control?
- Lots of tracking means too much work
- Too little focus
- More project tracking supports
  - Greater task switching …
    - Leads to more partially finished work …
      - Leads to more waiting …
        - Encourages more features "just in case" …
- To do: Be wary when adding management activities and tracking software.

Software In Process (SIP)

- Average days between specifying a feature and its availability

- From Requirements
- To Deployment in Production

SIP example 1

- Marketing asks development for Feature Q.
- Development produces an installation CD four months later
- SIP ~= 120

What does SIP tell us?

- Rollup metric indicating levels of
  - Value
  - Waste
- Low SIP shows ability to deliver real value quickly
- High SIP not necessarily bad

SIP example 2

- 3 applications into production
- Web portals
- *very complex employee transition management*
- Delivers every 2 weeks to production

- SIP ~10
  - *One Defect in last 9 months*

SIP example 3

- www.lifeware.ch
- 4000 tests run with every change
- Changes go into production every evening
- Only needed workflows implemented
- Policy redemption not coded until requested
- Low cost of operation
- SIP = 1

To do

- Calculate your project SIP
- What would change to cut that in half ?
- Which types of Waste are most obvious ?

Waste

- Partially Finished Work
- Extra Features
- Extra Process Steps
- Task Switching
- Waiting
- Motion
- Defects
- Management Activities

Tools* (topics in the series)

- Seeing Waste
- Value Stream Mapping
- Feedback
- Iterations
- Pull Systems
- Queuing Theory
- Cost Of Delay
- Systems Thinking
- Parallel Problem Solving
- Empower the team
- Motivation
- Leadership
- Expertise
- Perceived Integrity
- Conceptual Integrity
- Unit Testing
- Fitness Testing
- Refactoring

** Lean Software Development, by Mary Poppendieck*

See the Waste …

Be the Value !!!

- References:
  - Lean Software Development
    - Mary Poppendieck
  - Managing Software For Growth
    - Roy Miller
  - Extreme Programming Explained
    - Kent Beck

1-day Workshop Outline

Morning

- Why a Software Process
- What is waste? Value?
- [Case Study: Recognizing Waste & Value]
- The 4 Values of XP
- Process overview: A day in the life of XP
- Practices of XP

  Customer Practices

  Team Practices

  Programmer Practices

  Industrial Practices

(one of the following as time allows)

1- Growing Complex Systems

2- Case Studies

3- Limits of XP

Afternoon

- Charters and Retrospectives
- [Exercise] Write Project Charter
- Iteration & Release Planning

Development/Manufacturing Planning

- [Exercise] Planning
- [Exercise] Iteration #1
- [Exercise] Retrospective #1
- [Exercise] Iteration #2
- [Exercise] Retrospective #2

(as time allows)

- Overview of Common Tools:

  RUP XP-Plug in

  Fitnesse, xUnit, Refactoring, Code control, Auto Build, wiki