One Extreme Day

Extreme Programming In One Day

Mark Windholtz

ObjectWind Software Ltd

Who Am I

- Mark Windholtz
- ObjectWind Software Ltd
- 17 years Building Professional Software
- Business, Retail, Medical, Military, Financial, etc.
- First contact with XP in Oct 1996
- Started Six XP teams
- Consulted for Nine other XP teams

The Plan for the day

- 30m - **Introduction & Goals**
- 60m - Concepts
- 60m - Day in the Life of XP
- 60m - Demo: Test First Programming
- 45m - Lunch
- 60m - XP Planning
- 90m - Exercise: Story Writing / Release Planning
- 90m - Demo: Refactoring
- Wrap-up

Introduction

- Instructor introduction
- Classroom specifics
- Room access
- Restroom facilities
- Start and end times
- Eating facilities
- Equipment & materials

Target Audience

- Business Process Engineers: Responsible for confirming a project team's understanding of existing business processes and identifying requirements of the future system.
- Application Developers/Purchasers: Responsible for the analysis, design, selection, implementation, testing and/or deployment of the system.
- IT Software Project Managers: Responsible for life-cycle project management, communicating to stakeholders & the project resources. Customers are welcome

Goals

- Introduction to Extreme programming
- Business Case for XP
- XP Practices and Values
- Planning in a changing world
- Observe Test-first and Refactoring
- Experience Iteration Estimation, Planning, and Delivery.

P&G-Endorsed Methodologies

- Extreme programming (XP)
- Rational Unified Process (RUP)

Class Introductions

- Attendee introductions
  - name,
  - location,
  - position
- Background and interests
- Attendee methodology experiences

The Plan for the day

Two hydrogen atoms walk into a bar.
One says, "I've lost my electron."
The other says, "Are you sure?"
The first replies, "Yes, I'm positive..."

- 30m - Introduction & Goals
- **60m - Concepts**
- 60m - Day in the Life of XP
- 60m - Demo: Test First Programming
- 45m - Lunch
- 60m - XP Planning
- 90m - Exercise: Story Writing / Release Planning
- 90m - Demo: Refactoring
- Wrap-up

Agenda - Concepts

- Standard Practices
- Agile
- The Business Case
  - Lean Software Development
- Extreme Programming

Standard Industry Practice

- Most Software development is chaotic or bureaucratic
- "Code and Fix"; or wait while doing *other* tasks.
- Indicators
  - Long Integration Phase
  - Long Test Phase
  - High Defect Counts

Solution of the 1990's

- Rigorous Heavy Process
- Lots of Documents
- Lots of Control Points
- Blue Prints before you build

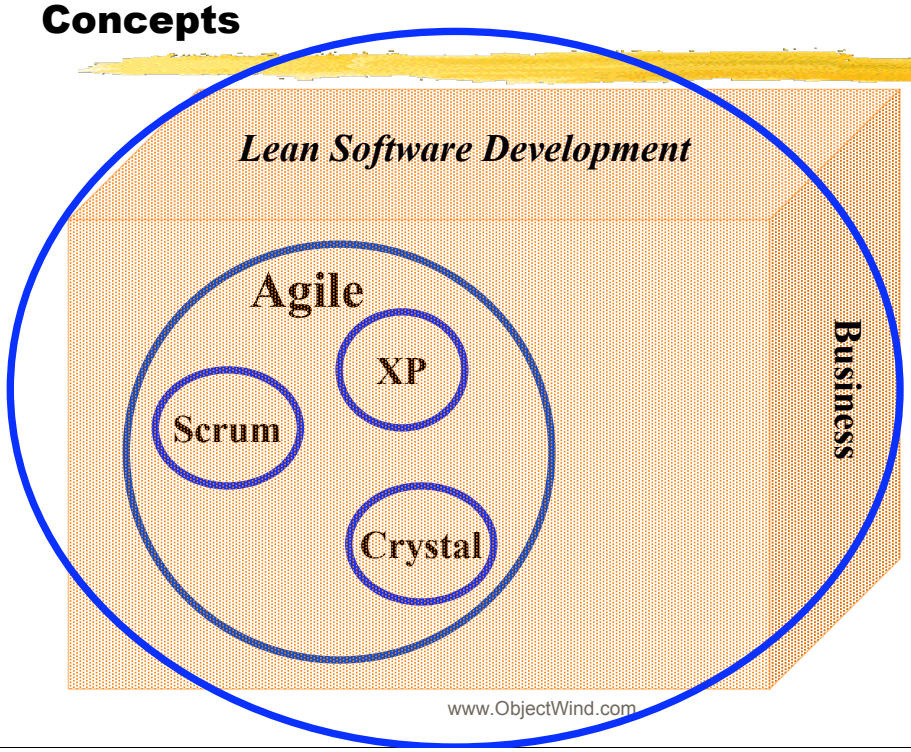
Why not common practice?

- Not Very Successful
  - *if only our people would follow this great process!*
- Conflicting Goals
  - Process vs. Product
- Criticized as bureaucratic & complicated
  - *What's harder: building the s/w or implementing RUP?*
- Big Processes often introduce added organizational conflict

Extreme in the Middle

- Compromise
  - Between No Control and No Movement
  - Between Chaos and Over Complicated

Concepts



Concepts

- Business Model
  - Lean Software Development
- Approach
  - Agile
- Technique
  - Extreme Programming, Scrum, etc …

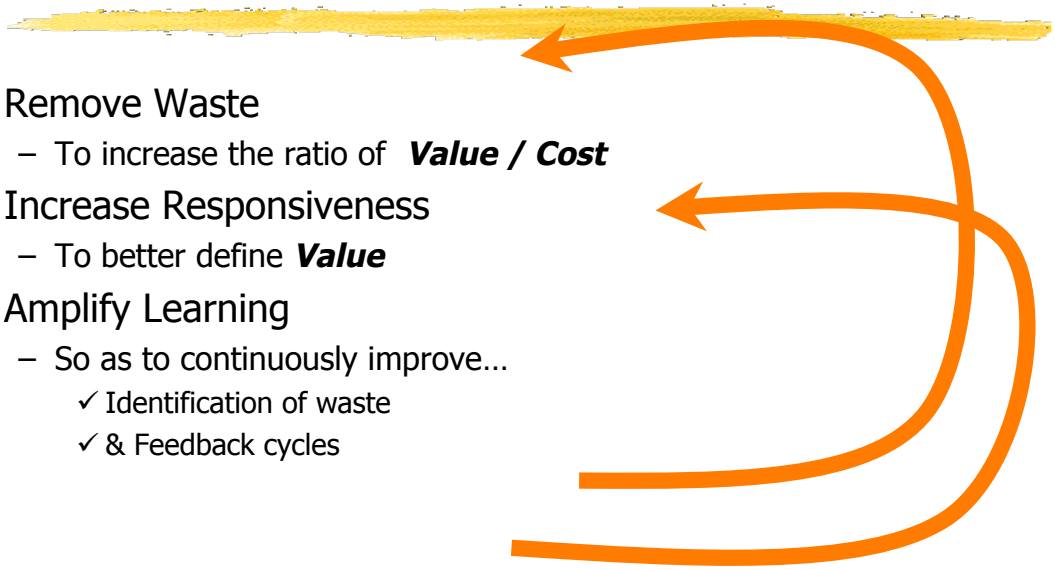
Lean Thinking Companies

- Fed Ex
- LensCrafters
- SouthWest
- Toyota

< Lean Concepts >

- Value
  - What the Customer will pay for
- Pull
  - Respond when the Customer asks
  - Not when your batch of deliverables is ready
- Flow
  - Move value smoothly, continuously
- Perfection
  - Quality is Free
  - Continuous Improvement

Lean Goals

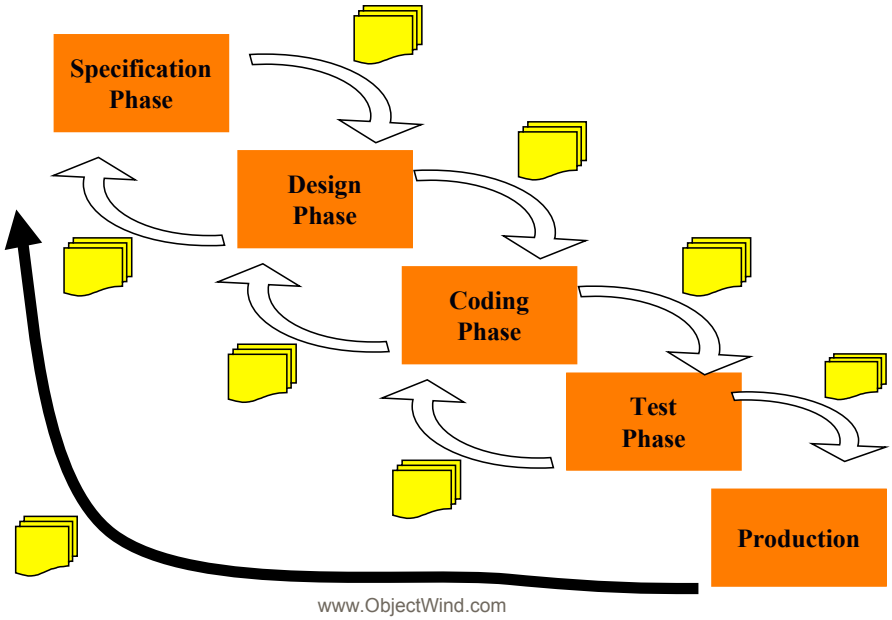


- Remove Waste
  - To increase the ratio of ***Value / Cost***
- Increase Responsiveness
  - To better define ***Value***
- Amplify Learning
  - So as to continuously improve…
    - ✓ Identification of waste
    - ✓ & Feedback cycles

Inventory is Waste

- Inventory is a non-producing Asset
- Inventory reduces responsiveness
- Any Artifact not producing Current Income
- Features under development are Inventory
  - ✓ Don't know **if/how** it will work
    - until it **does** work (Risk)
  - ✓ High Administrative costs
  - ✓ Reduced response to changing markets
  - ✓ Higher (Longer) Return on Investment
    - Consider: Net Present Value
  - ✓ Time between Idea and Market feedback

Phases Lead to no-value Artifacts



What else is Waste?

- Inventory
  - Software In Process (SIP)
- Defects not caught
- Over -production, elaboration
- Extra Process Steps
- Motion getting the needed information
- Waiting for information
- Transportation - handoffs

Example 1

- 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*

What's your top Problem?

- That's Easy!

- Typical s/w project 6-12 months
- Quiz:
  - One year from now,
  - what will be your top 10 business problems?

- Ummm ??

Just In Time Software

- Let Customer choose Value
- Reduces waste of Over-Engineering
- Deep Focus on quickest payback

Most Important feature	Most Important feature	Most Important feature	Most Important feature
Specify	Specify	Specify	Specify
Design	Design	Design	Design
Code	Code	Code	Code
Test	Test	Test	Test
Deploy	Deploy	Deploy	Deploy

Example 2

- 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

< XP Values >

- Communication
  - visibility
- Simplicity
  - don't solve problems you don't have
  - Remove Waste
- Feedback
  - define short term success/failure criteria
  - Start with the end in mind
- Courage
  - Skiing advice: "lean forward"
  - Communication + Simplicity + Feedback => Courage

Q: What's the difference between going to jail and installing a new Microsoft product on your computer?
A: When you go to jail you get one free phone call.

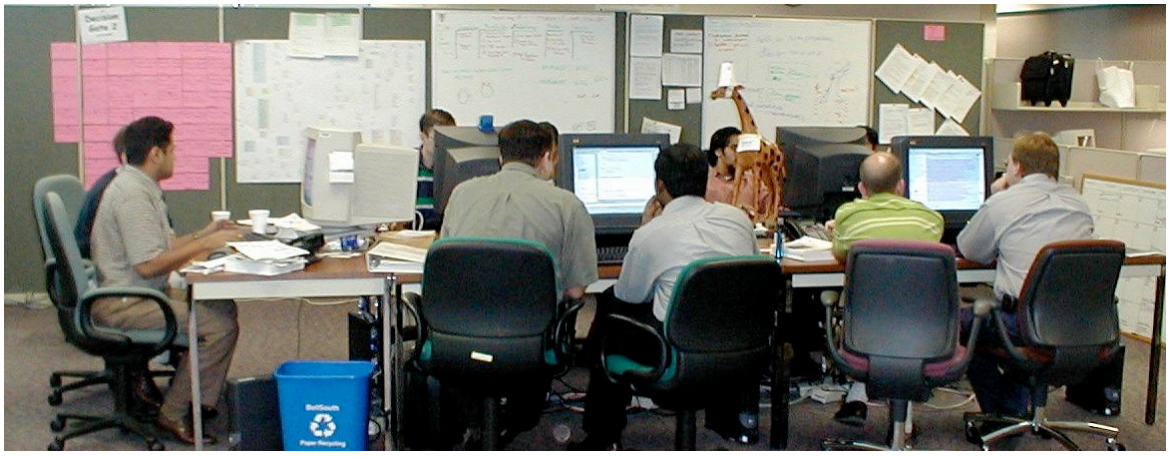
The Plan for the day - (One Day)

- 30m - Introduction & Goals
- 60m - Concepts
- **60m - Day in the Life of XP**
- 60m - Demo: Test First Programming
- 45m - Lunch
- 60m - XP Planning
- 90m - Exercise: Story Writing / Release Planning
- 90m - Demo: Refactoring
- Wrap-up

< A Day in the life of XP >

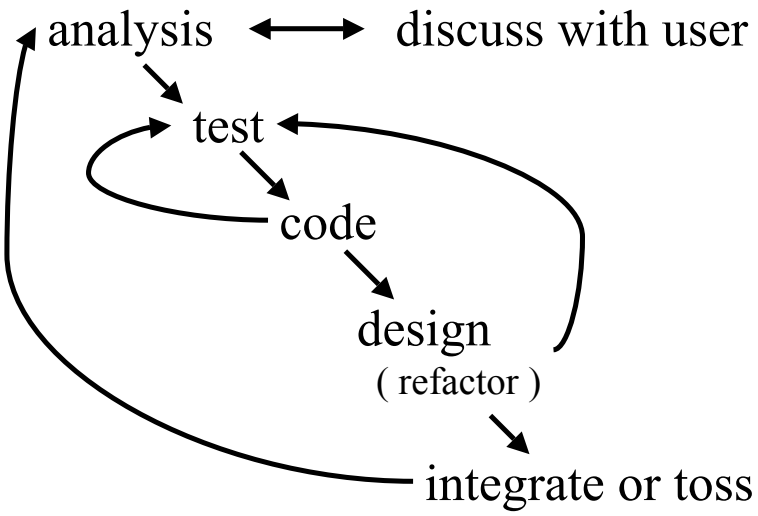
- 15 minute team stand-up meeting.
- Pair-up
- Go to the open workspace
- Get a story from task board
- Talk with the customer sitting with the team
- Engage in programming episodes
- Monitor the automated build & test system
- Deliver a few fully tested, integrated business Stories
- Go home at a reasonable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

Open Workspace



The Task Board

XP Episode Cycle



< Core Practices >

Customer	Programmer	Team
How to Define Features Iteratively	How to Build Features Iteratively	How to Communicate & Schedule
Planning Game	Test-First	Open Workspace
On-site Customer	Simple Design	Continuous Integration
Acceptance Tests	Pair Programming	Collective Ownership
Small Releases	Coding Standard	Sustainable Pace
	Continuous Design (Refactoring)	Ubiquitous Language (Domain Model)

Customer Practices - 1

- Planning Game

  Programmers estimate, Customer chooses stories

  - ***Game*** means optimizing available resources
  - Two types: Release planning & Iteration planning
  - Adapt the plan to reality (not vise versa)

- On-site Customer

  Customer representative sits with the team

  - Separates business decisions from technical decisions
  - Allows team to proceed quickly without fear of missing info

Customer Practices - 2

- Acceptance Tests

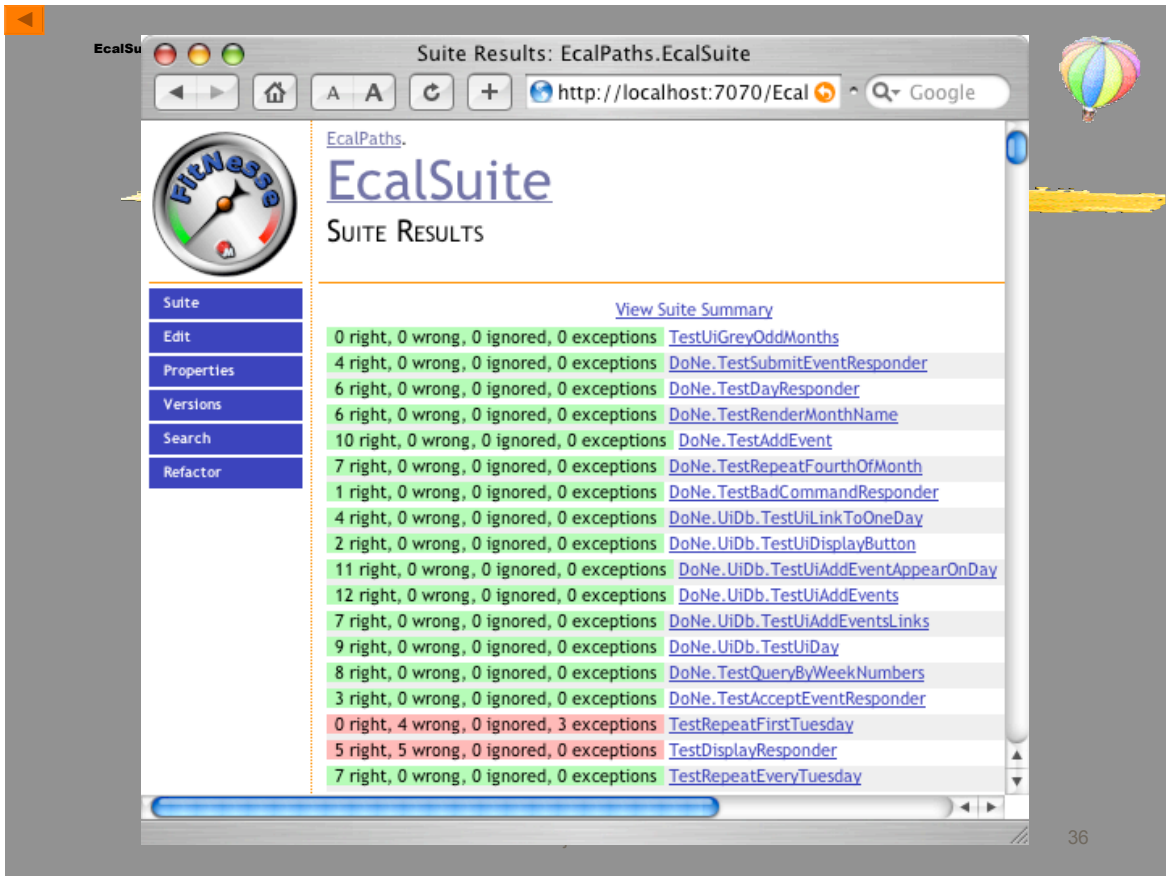
  Automated end-to-end testing of Customer specified features

  - Customer determines when a story is correct
  - Allows continuous regression testing

- Small Releases

  Go live as quickly and often as possible (4-weeks, every day?)

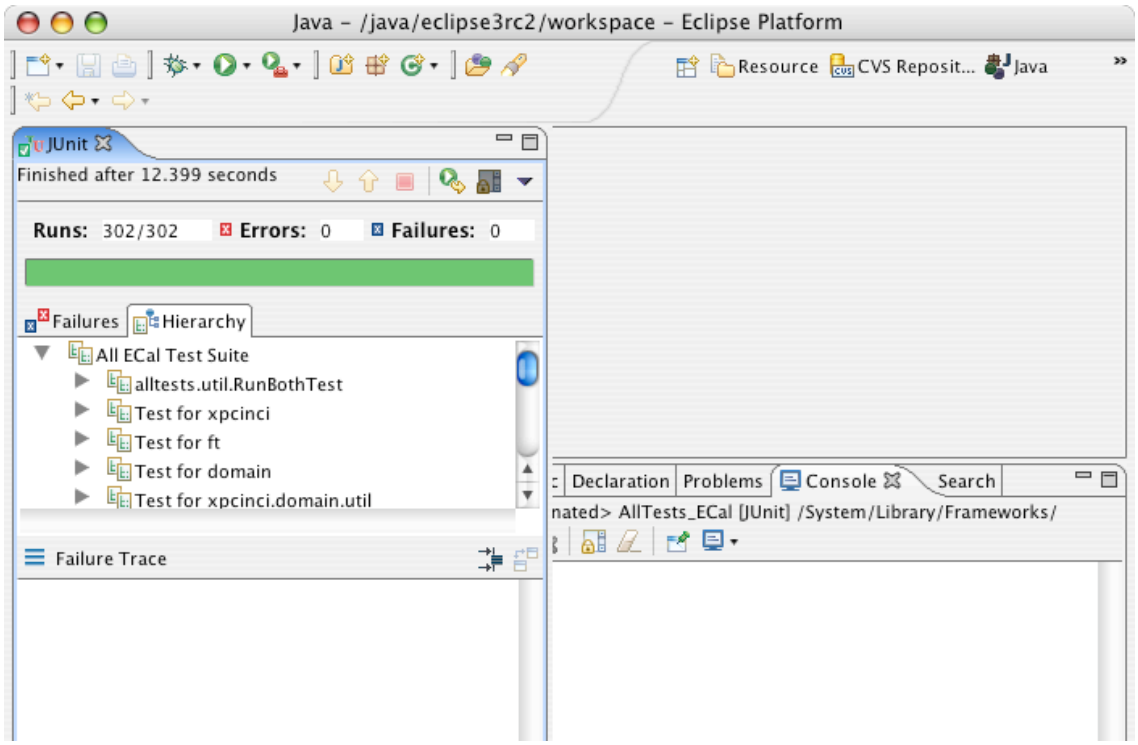
  - Quicker feedback from customer
  - Smaller ratchet steps forward
  - Instills the discipline of delivery & deployment



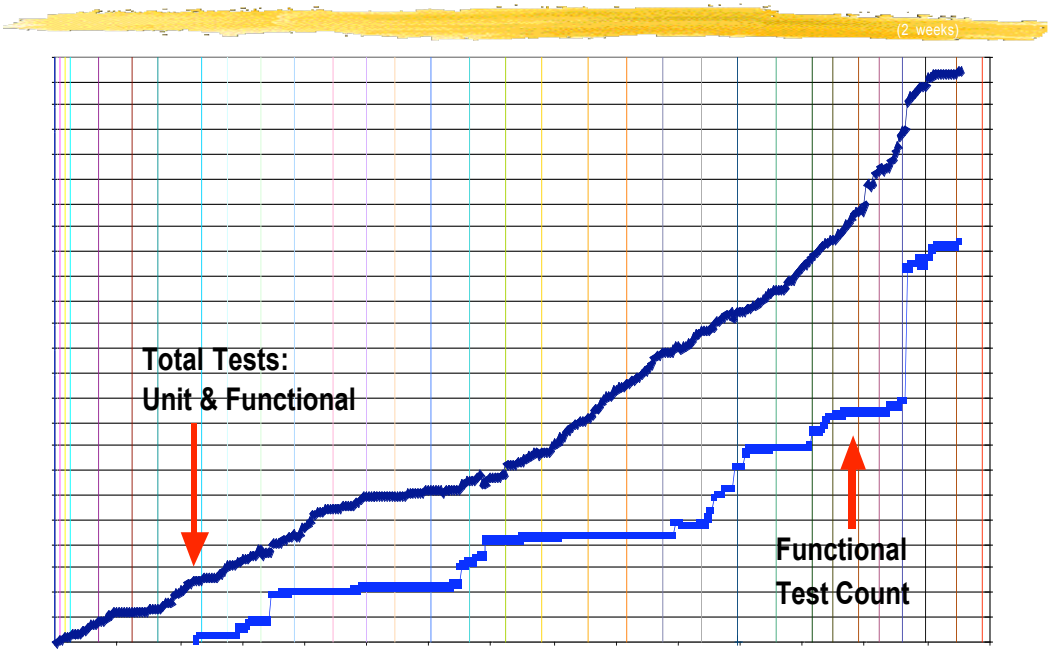
Programmer Practices - 1

- Test-First
  All production code is written after a failing test
  - Start with the end in mind
  - Defines the task completion criteria
  - Defines the interface
  - Supports Continuous Integration
- Simple Design
  Design the simplest thing that could possibly work
  1. Passes all tests
  2. Expresses all concepts clearly
  3. No Duplicate code
  4. Fewest Classes and Methods
  - Don't solve problems you don't have
  - Easier to understanding, maintain, and extend
  - Eliminates Over-Engineering

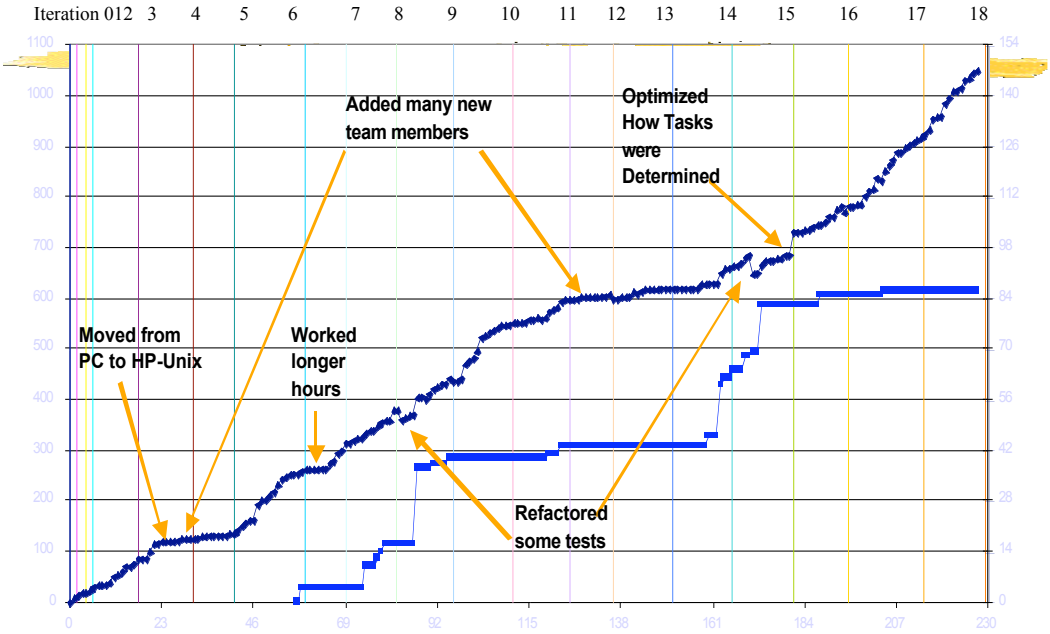
JUnit TestRunner in Eclipse 3.0



Unit & Functional Tests



Unit & Functional Tests w/ Notes



Programmer Practices - 2

- Coding Standard

  Code should look similar across whole the system

  - Code is quickly understood
  - Code is quicker to write
  - No code castles

- Continuous Design (Refactoring)

  Continuously Eliminate duplicate code

  - Allows for Micro-design
  - Eliminates the waste of duplicate code
  - Supports the growth of the Domain Model
  - Increases code reuse
  - Decreases effort in fixing defects
  - Decreases effort in changing code when features tomorrow

Programmer Practices - 3

- Pair Programming

  All production code written by two programmers on one keyboard

- Code review is good,
- So review all code as it is typed in
- One person types (tactical)
- One person thinks (strategic)
- Roles switch continuously
- Fewer defects
- Reduces unneeded tasks
- Increases spread of understanding
- Reduces bottlenecks

Pair Programming Photos

Pair Programming

Team Practices - 1

- Open Workspace

  Team sits together in the same room
  - Remove barriers to communication
  - It's not noise if it's related to the same project
  - Allows valuable Incidental Communications
  - Best flow of information

- Continuous Integration

  Integrate Code Often (every 1-2 hours)
  - Small Integrations are painless
  - Eliminated need to branch and merge the source tree
  - Reduces task dependencies to minimum
  - Increases visibility of project progress
  - Allows automated build and test to run continuously

Auto Build

- Build tools
  - Cruise Control
  - Anthill
- What it does
  - Check out of source control
  - Build code on clean machine
  - Run unit tests
  - Run Acceptance tests
  - Run metrics
  - Version stamps passing code
  - Email report of unexpected results
  - Details available on a web interface

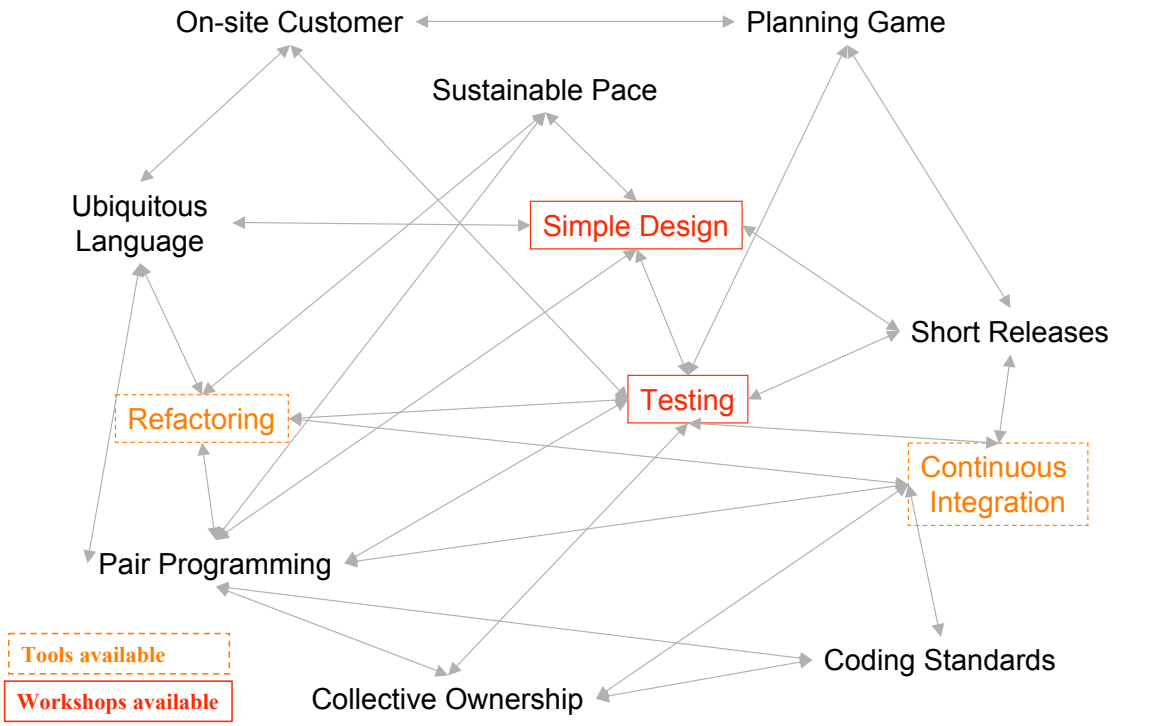
* Cruise Control Example

- << insert screen capture >>

Team Practices - 2

- Collective Ownership
  - Removes code castles
  - Reduces Information Hoarding (no squirrels)
  - Encourages coding standard
- Sustainable Pace
  - Increases accuracy of planning
  - Reduces defects from burned out programmers
- Ubiquitous Language
  - Metaphor (original)
  - System of Names
  - Domain Model, Ubiquitous Language (Eric Evans)
  - Conceptual Integrity (Fred Brooks)

Practices Support Each Other



Simple Process

- People easily know what they should do.
- People talking together, and learning.
- Tested and running clean code.
- Short term measured goals.
- Defined Completion and Quality
- Focus on Sustainable, Repeatable, Deployment

Quality code is faster to produce

- Quality
  - Pair Programming
  - Unit testing, Acceptance Testing
  - Anyone can clean the code
- Less Backtracking
  - 80% Less debugging
  - Dramatically Less Defects
- Clean, Tested code is easier to change than a UML model

What did the zero say to the eight?
Nice belt.

The Plan for the day

- 30m - Introduction & Goals
- 60m - Concepts
- 60m - Day in the Life of XP
- **60m - Demo: Test First Programming**
- 45m - Lunch
- 60m - XP Planning
- 90m - Exercise: Story Writing / Release Planning
- 90m - Demo: Refactoring
- Wrap-up

< Demo: Test First >

- Ecal is a web based Event Calendar application being built Open-Source for Children's Hospital by the xp-cinci User's Group.
  (xp-cinci is Extreme-Programming users group of Cincinnati, www.objectwind.com/xp-cinci)
- add a new feature to ECal.
- Event scheduling on a day-of-month.
- You will see:
  - Acceptance Testing with FitNesse to set completion criteria
  - Unit testing with jUnit, failing tests then successful tests
  - How tests lead to the next logical implementation step

A sandwich walks into a bar.
The bartender looks and him and says,
"Hey, we don't serve food here."

The Plan for the day - Lunch

- 30m - Introduction & Goals
- 60m - Concepts
- 60m - Day in the Life of XP
- 60m - Demo: Test First Programming
- **45m - Lunch**
- 60m - XP Planning
- 90m - Exercise: Story Writing / Release Planning
- 90m - Demo: Refactoring
- Wrap-up

Q: Whats black and white and moos like a cow?
A: A Cow.

Werner Heisenberg is pulled over by a policeman who says, "Do you know how fast you were going?" Heisenberg answers, "No, but I know exactly where I am.

The Plan for the day - Planning

- 30m - Introduction & Goals
- 60m - Concepts
- 60m - Day in the Life of XP
- 60m - Demo: Test First Programming
- 45m - Lunch
- **60m - XP Planning**
- 90m - Exercise: Story Writing / Release Planning
- 90m - Demo: Refactoring
- Wrap-up

Delivering to Deadline

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

Need for Planning

General Dwight D. Eisenhower said

> In preparing for battle, I have always found that plans are useless, but planning is indispensable.

- Don't let the plan become more central than reality
- Management is about building realistic plans, and adjusting the plan as reality changes

What Management Gets

- Management gets …
  - An Overall release plan (What, when, and how much)
  - Most value out of each programming week
  - Visible Progress: code running repeatable tests
  - To Change direction as business changes
  - Be informed early of schedule changes

What Programmers Get

- Programmers are …
  - told what is needed in priority order
  - Allowed to produce quality work
  - Allowed to ask for help
  - Allowed to make and update estimates
  - Allowed to accept responsibility

Business to Code Process

Product is a bunch of …

Release is a bunch of …

Iteration is a bunch of …

Acceptance Tested Stories is a bunch of …

Episode is a bunch of …

Test is a bunch of …

Code

Time Scale

- Product: Years
- Release: Month(s)
- Iteration: 1-2 Week(s)
- Acceptance Test Story : 1-3 days
- Episode: 1-3 hours then integrate
- Test: 1 to 5 minutes
- Code: 0 to 20 minutes to satisfy test case

User Story

- Written on an index card by the Customer
- Simple title to remind everyone of the feature
- Business focused feature
- Promise to continue the conversation to get more detail later
  - ✓ Information Just-In-Time
- Story completion is checked by an Acceptance test

Four operations on a User Story for planning

1. Discuss it until enough detail to estimate
2. Split it if it's too big
3. Spike it if the technology is not understood
4. Estimate it

Example User Stories

- Student purchase monthly parking pass online.
- Accept credit card payment
- Accept PayPal ™ payment
- Professor input student grades
- Student view of current seminar schedule
- Student order of official transcripts.
- Students can only enroll in seminars for which they have prerequisites
- Transcripts will be available online
- Support for browsers: IE, FireFox, &Safari

What is Progress?

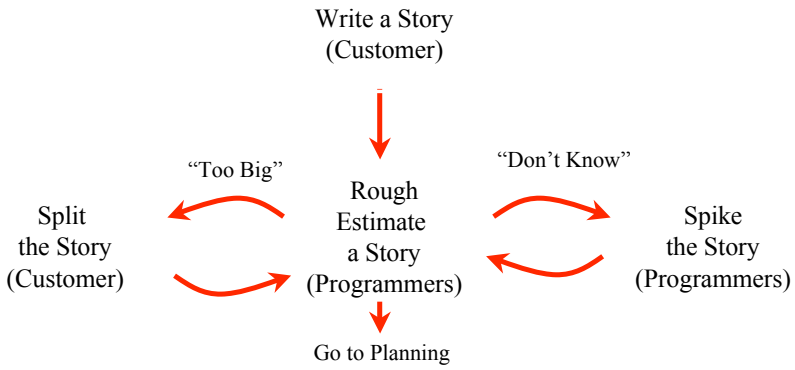
- A Story is either **Done** or **Not Done**
  - *70% done* is meaningless
- Done means done!
  - *Done but not tested* is meaningless
  - *Done but not integrated* is meaningless
- Code is written in small, running, tested pieces
- No large leaps forward in a fog …
  … and falling off a cliff

Velocity

- Stories are estimated
  - 1 to 3 *ideal days or hours*
  - Measured in relative *story points*
- How many story points were delivered?
- Historical velocity data
  - Rough (but real) predictor future progress
  - Cost, scope, schedule

Release Planning Game

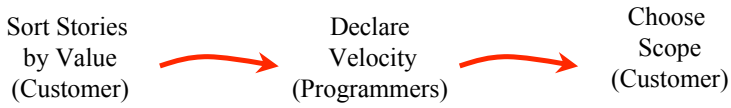
(Exploration)



- Goal
  - Understand what the system is to do
- Result
  - A working set of estimates

Release Planning Game

(Planning)

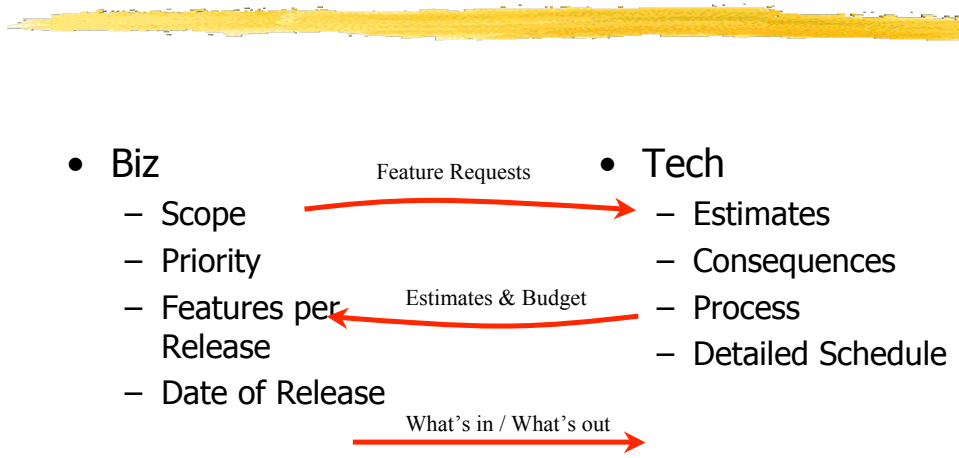


- Goal
  - Plan the next Release so it gives the most bang for the buck.
- Input
  - Set of estimated stories
- Result
  - A prioritized list of stories currently planned to by included in next release

Release Planning

Iteration Planning Game

(finding a fit)



Iteration Planning Activities (4-8 hours)

- Large meeting room with white boards and room to pace
- Snacks and toys provided
- Begin with Retrospective
  - Review of previous iteration
  - Calculate previous iteration velocity results
  - Improvement plan
- Customer provides stories for iteration
- Discussion
- Spike (experiments)
- Initial High level design of each Story as needed
- Estimation of Stories
- Splitting Large Stories
- Programmers sign-up and commit to individual stories

Iteration Planning

Iteration Planning

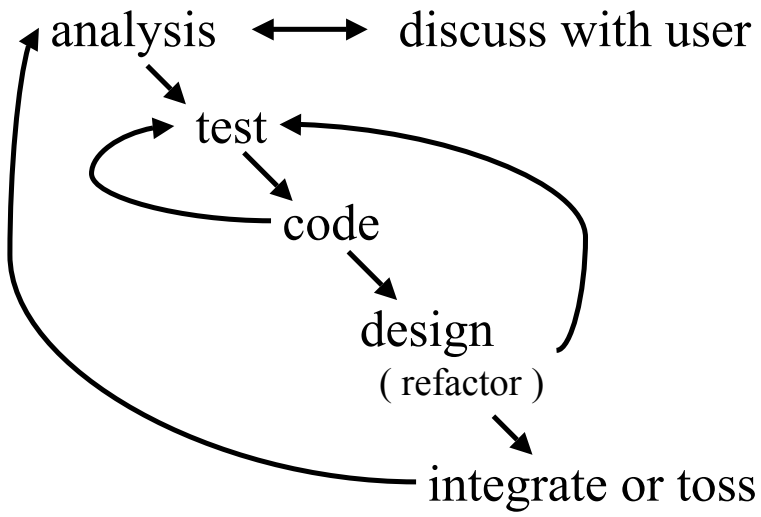
Iteration Planning Results

- List of stories and tasks for the iteration
- Basic design for each story is understood
- Estimate for each story has been committed to by a programmer
- Story cards are posted on the Task board with the initials of the programmer
- Programmers can begin coding the Stories
- Customer can begin defining the Acceptance tests for those stories

Time Scale

- Product: Years
- Release: Month(s)
- Iteration: Week(s)
- Acceptance Test: 1-3 Days
- Episode: 1-3 hours then integrate
- Test: 1 to 5 minutes
- Code: 0 to 20 minutes to satisfy test case

XP Episode Cycle



A neutron walks into a bar and orders a drink.
The bartender says: "for you, no charge".

The Plan for the day - Exercise

- 30m - Introduction & Goals
- 60m - Concepts
- 60m - Day in the Life of XP
- 60m - Demo: Test First Programming
- 45m - Lunch
- 60m - XP Planning
- **90m - Exercise: Story Writing / Release Planning**
- 90m - Demo: Refactoring
- Wrap-up

Q: What did the cannibal say about the clown?
A: "He tastes funny."

Story Writing Exercise: 90 minutes

- Project Charter: Build a better mouse trap
- Split into a customer, & developers
- Customer writes stories, and acceptance tests
- Developers estimate, design, and implement (by drawing) a solution.
- Customer runs acceptance tests on the solution

A neutron walks into a bar and orders a drink.
The bartender says: "for you, no charge".

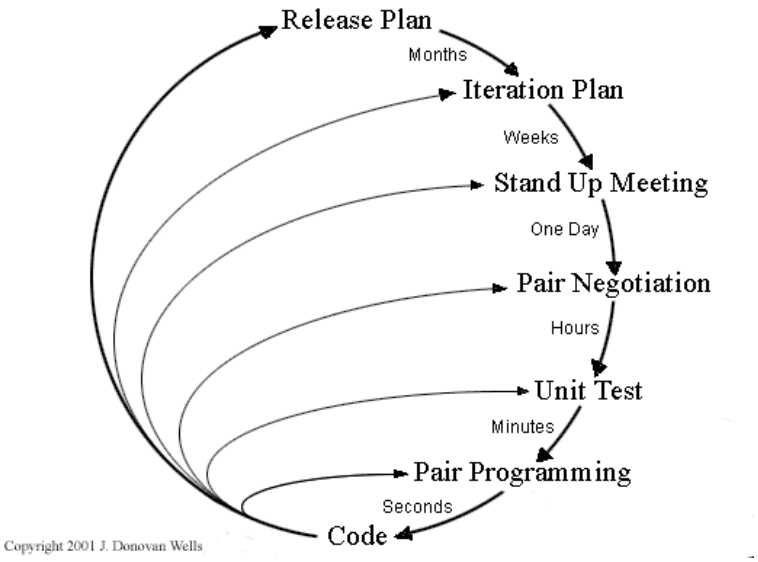
The Plan for the day - Wrap

- 30m - Introduction & Goals
- 60m - Concepts
- 60m - Day in the Life of XP
- 60m - Demo: Test First Programming
- 45m - Lunch
- 60m - XP Planning
- 90m - Exercise: Story Writing / Release Planning
- 90m - Demo: Refactoring
- **Wrap-up**

Continuous Feedback

- Continuous…
  - Code reviews (pair programming)
  - Unit testing
  - Integration testing
  - User acceptance testing
- Feedback through estimation, testing, short releases
  - Gives user confidence
  - Gives developers confidence
  - Gives Management confidence
  - Growing Software (iterative process)
- Use feedback to update the Plan

Feedback Loops



Continuous Best Practices -1

- If design is good, make it everyone's job (Refactoring, domain design)
- If simplicity is good, use the simplest design that supports the current features (simple design)
- If delivering code is good, make iterations really short (days instead of months)? (short releases & acceptance testing)

Continuous Best Practices - 2

- If code reviews are good, do it all the time (pair programming)
- If testing is good, do it all the time…even customers? (continuous unit and acceptance testing)
- If integration testing is good, do it several times a day? (continuous integration)

Transition to XP

- Culture shift can be difficult for Developers and Managers
- Requires experienced OO developers (but not gurus)
- May not get all practices, all the time
- A good XP coach can help guide team to
  - Work with what you've got
  - Add practices as needs appear
  - Increase testing and OO design skills
  - Diffuse common interpersonal conflicts
  - See how the practices save effort and time
  - Keep the team on-task in a changing environment

Challengers to XP success

- Additional practices may be needed when attempting
  - Rigid organizations (controls instead of goals)
  - Fearful organizations (shoot the messenger)
  - Rewards based on input rather than results
    - ✓ The time spent at work is primary measure of employee value.
  - Prima donna developers
    - ✓ Sub-species: Information Hoarders (squirrels)
  - Wrong physical environment (isolated cubes)
  - Large teams over 35 [16]
  - Distributed "team" members
  - See: www.industrialxp.org

XP and the wider world

- Documentation?
  - Focus on cleaner, unit tested, peer reviewed code.
  - Other documentation as needed (consider a wiki).
  - Be aware that it is an expense
- XP & UML?
  - Use UML if it helps. Be aware that it is an expense
  - Don't let "pictures" and models bog you down
  - Don't be afraid of reading the code
  - See "Agile Modeling" [14]
- XP & RUP
  - RUP provides an XP Plug-in
- Does XP work within context of CMM?
  - XP practices map to CMM processes [12,13]

< XP & RUP >

Characteristics	Extreme Programming	Rational Unified Process
Project Size	Small to medium	Medium to large
Delivery time	Need quick delivery	Can afford to take longer, developing key modules
Number of development resources	Up to a 20 programmers	More than a 20
Programmer Approach	Programmers work together in same room, and contribute their entire skill set.	Geographically dispersed programmers. May have a need for role separation.
Code Review Approach	Pair programming, Test-First, aggressive Refactoring, continuous Integration, automated builds.	Code reviews with more than two programmers. Accepts code duplication.
Iteration Size	End to end, iterations may be 1, 2 or 3 weeks long	Iterations may be longer than 3 weeks.
Release Approach	Characterized by small and often releases. Start with small, useful feature set. Customers are available to accept multiple releases.	Releases can be planned out. May include larger module releases. Customer may want single release to minimize user impact.
Project Documentation	The code and tests are the documentation. Other models as needed.	Allows for visual models. Models may be used to generate downstream code.

There are 10 kinds of people in this world. Those who read binary numbers and those who don't...

References - 1

1. XP Cincinnati, Users Group www.objectwind.com/xp-cinci
   Local group to explore tools, techniques, and practices of XP
2. Beck, Kent. **Extreme Programming explained**, 2000
3. Fowler, Martin. "The New Methodology". Nov 2001.
   www.martinfowler.com/articles/newMethodology.html
4. Jefferies, Ron. "They're called practices for a reason", Sept 2000.
   www.xprogramming.com/xpmag/PracticesForaReason.htm
5. www.junit.org - testing frameworks
6. www.refactoring.org
7. Fowler, Martin. **Refactoring: Improving the Design of Existing Code.** Addison-Wesley. 2000
8. www.pairprogramming.com - resources
9. www.extremeprogramming.org - on-line methodology manual
10. www.xprogramming.org - many articles
11. www.objectWind.com - articles on Lean Software Development and XP

A thief broke into the local police station and stole all the toilets. A spokesperson was quoted as saying, "The police have absolutely nothing to go on."

References -2

12. Paulk, Mark (SEI). "XP from a CMM Perspective". Invited Talk. XPUniverse, July 2001. Raleigh, NC.
13. http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap - The Extreme Programming Roadmap on the Portland Pattern Repository. Where most of the discussions and good information take place in Wiki-style.
14. www.agilemodeling.com – Scott Ambler's site
15. www.agilealliance.org - Non-profit organization dedicated to promote agile methodologies
16. *Agile Software Development in the Large - Diving Into the Deep*, by Jutta Eckstein; Dorset House, New York, 2004 ISBN 0-932633-57-9
17. The Central Florida Patterns Group
    http://groups.yahoo.com/group/cflpatternsgroup/
18. Industrial XP, additional Practices for large scale XP - www.industrialxp.org

References -3

P&G Internal Web Sites

Sys Dev Competency Web Site : http://edlecom1.na.pg.com/sysdevcomp/default.htm

Sys Dev Competency COP Teamspace: http://sysdevcomp.internal.pg.com/

Java Community Of Practice Teamspace: http://javacop.pg.com/

.NET Community Of Practice Teamspace: http://dotnetdev.pg.com/

External Web Sites

Articles: www.xprogramming.org

Articles: www.objectmentor.com/resources/articleIndex

Extreme Programming: www.extremeprogramming.org

Agile Modeling: www.agilemodeling.com

Modeling: www.modelingstyle.info

Agile Alliance: www.agilealliance.com

Unit Testing: www.junit.org

Conferences: www.xpuniverse.com

References - 4

Some Useful Books

Extreme Programming Explained: Embrace Change by Kent Beck

Planning Extreme Programming by Kent Beck, Martin Fowler

Agile Software Development, Principles, Patterns, and Practices by Robert C Martin

Agile Software Development by Alistair Cockburn

Agile Modeling: Effective Practices for Extreme Programming and the Unified Process
by Scott W. Ambler, Ron Jeffries

Writing Effective Use Cases by Alistair Cockburn

Lean Software Development (An Agile Toolkit) by Mary Poppendieck

Refactoring: Improving the Design of Existing Code by Martin Folwer, Addison-Wesley. 2000