

# Planning for Web Applications

Webmasters' Guild

March 5, 2004

A stylized, dark teal silhouette of a mountain range is located in the bottom right corner of the slide, extending from the right edge towards the center.

# Agile Software Development


- ◆ A philosophy of software development with these principles:
  - **Individuals and interactions** *over* processes and tools
  - **Working software** *over* comprehensive documentation
  - **Customer collaboration** *over* contract negotiation
  - **Responding to change** *over* following a plan

# Agile Software Development


- ◆ Works best:
  - for small to medium projects
  - when customers buy in
  - when IT staff are true believers
- ◆ Fails when:
  - we fall back into old habits
  - we try to launch with a Big Bang
  - customers don't work with us

# Agile Software Development

## ◆ Goals

- Satisfy the customer
  - Embrace changing requirements
  - Deliver working, useful software every few weeks
  - Business people and developers work together
  - Working software is the primary measure of progress
- 

# Goals of Project Management

- ◆ Planning
  - ◆ Estimation
  - ◆ Predictability
  - ◆ Tracking
  - ◆ Quality
- 

# Rules and Practices



## The Rules and Practices of Extreme Programming.

*Lessons  
Learned*

### Planning

- ✎ ✎ [User stories](#) are written.
- ✎ ✎ [Release planning](#) creates the schedule.
- ✎ ✎ Make frequent [small releases](#).
- ✎ ✎ The [Project Velocity](#) is measured.
- ✎ ✎ The project is divided into [iterations](#).
- ✎ ✎ [Iteration planning](#) starts each iteration.
- ✎ ✎ [Move people around](#).
- ✎ ✎ A [stand-up meeting](#) starts each day.
- ✎ ✎ [Fix XP](#) when it breaks.

### Coding

- ✎ ✎ The customer is [always available](#).
- ✎ ✎ Code must be written to agreed [standards](#).
- ✎ ✎ Code the [unit test first](#).
- ✎ ✎ All production code is [pair programmed](#).
- ✎ ✎ Only one pair [integrates code at a time](#).
- ✎ ✎ [Integrate often](#).
- ✎ ✎ Use [collective code ownership](#).
- ✎ ✎ Leave [optimization](#) till last.
- ✎ ✎ No [overtime](#).

### Designing

- ✎ ✎ [Simplicity](#).
- ✎ ✎ Choose a [system metaphor](#).
- ✎ ✎ Use [CRC cards](#) for design sessions.
- ✎ ✎ Create [spike solutions](#) to reduce risk.
- ✎ ✎ No functionality is [added early](#).
- ✎ ✎ [Refactor](#) whenever and wherever possible.


### Testing

- ✎ ✎ All code must have [unit tests](#).
- ✎ ✎ All code must pass all [unit tests](#) before it can be released.
- ✎ ✎ When [a bug is found](#) tests are created.
- ✎ ✎ [Acceptance tests](#) are run often and the score is published.

[ExtremeProgramming.org](http://ExtremeProgramming.org) home | [XP Map](#) | [Email the webmaster](#)

Copyright 1999 Don Wells all rights reserved

# Lightweight Project Management

- ◆ When full-blown project management would double the cost of your project
  - ◆ When the team is small
  - ◆ When the project is small
- 

# Extreme Programming

- ◆ A lightweight project management method that supports agile software development

# Extreme Programming

- ◆ Today's session: XP planning techniques
  - User stories
  - Release planning
    - ◆ Small releases
    - ◆ Iteration planning
    - ◆ Project Velocity

# Extreme Programming

## ◆ User Stories

User Stories are written by the customers as things that the system needs to do for them.

They are in the format of about three sentences of text written by the customer in the customers terminology without techno-syntax.

# Extreme Programming

## ◆ Task 1

- Collect user stories

# Extreme Programming

## ◆ Task 2

- Organize user stories into broad functions

# Extreme Programming

## ◆ Task 3

- Eliminate duplication by negotiating with the customers

# Extreme Programming

## ◆ Task 4

– Estimate the user stories

– Rules:

◆ 1,2, or 3 “ideal” weeks

◆ If more than 3, then break out into new, more focused stories

◆ If less than 1, then paper clip into a batch

## ◆ Task 5

– Calculate the total system time

- ◆ Multiply the score by Project Velocity (we'll use 1.6)
- ◆ Divide by available technical staff

# Extreme Programming

## ◆ Task 6

– Plan the first release

- ◆ Agree on delivery date
- ◆ Determine the number of points that can be completed in that time
- ◆ Decide what user stories will be in the release

# Resources

- ◆ The AgileAlliance, a non-profit organization promoting the concepts of agile software development  
<http://www.agilealliance.com/home>
- ◆ Extreme Programming: A gentle introduction.  
<http://www.extremeprogramming.org>
- ◆ The Portland Pattern Repository (a Wiki – very rich)  
<http://c2.com/cgi/wiki?ExtremeProgrammingRoadmap>
- ◆ Principles behind the Agile Manifesto  
<http://agilemanifesto.org/principles.htm>
- ◆ Refactoring Home Page  
<http://www.refactoring.com>
- ◆ Development Advisor: An Introduction to Extreme Programming  
<http://www.advisor.com/doc/13571>