Extreme Programming
(3C05/D22)

UCL ComputerScience

Unit 6: Extreme Programming

Objectives
– To review extreme programming an alternative software system
development process rapidly gaining in popularity.

UCL ComputerScience

Extreme Programming

No!

Peter Line
Photo: Mark Gallup
Lake Louise, Alberta

UCL ComputerScience

Extreme Programming (XP)

- A deliberate and disciplined approach to software systems development
- Developed by Kent Beck
- About 4 years old, increasingly used in both large and small organisations
- Emphasises communication, feedback simplicity and above all customer feedback
- A member of a growing family of "lightweight methods"

ComputerScience

When to Use XP

- Remember you have to match the process to the problem!
- XP works when:
  - Requirements are changing rapidly
  - High risk, new challenge projects
  - Small groups of programmers (between 2-10)
  - Able to create automated tests
  - Direct customer involvement is possible

ComputerScience

Rules and Practices of XP

- 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

ComputerScience

User Stories

- User Stories are written by the customers
- Things that the system needs to do for them (similar to use cases)
- In the format of about three sentences of text written by the customer in the customers terminology
  - without techno-babble!
- About 80 +- 20 is a typical number for a mid-sized project, each story between 1 and 3 weeks of ideal development time
- User stories drive creation of acceptance tests

ComputerScience

Iterations and Releases

- Release plan sets out overall schedule
- User stories are estimated in terms of effort and then are written on cards and prioritised by customer
- User stories are then divided into releases by development team
- Divide development into about 12 iterations of about 1 to 3 weeks in length and have an iteration planning meeting before each iteration
- Project velocity=number of user stories per iteration
- Release often and aim to deliver a useable, testable system early

ComputerScience

Rules and Practices of XP

- 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.

ComputerScience

Design

- Never add functionality before it is required!
- Organise your design around a shared model (metaphor). Work hard to create a simple understandable design.
- A spike solution is a very simple program to explore potential solutions. Build a system which only addresses the problem under examination and ignore all other concerns.
- Refactor mercilessly! Refactoring is the removal of redundancy, elimination of unused functionality, and rejuvenation of obsolete designs in order to keep the design simple and avoid needless clutter. It helps to make a design easier to understand, modify, and extend.

ComputerScience

CRC Cards

- CRC cards are a useful tool
  - CRC- class, responsibilities, collaborations
  - Class at top of card, responsibilities listed down the left side, collaborating classes listed to the right of each responsibility
- A CRC session proceeds with someone simulating the system by talking about which objects send messages to other objects. By stepping through the process weaknesses and problems are easily uncovered. Design alternatives can be explored quickly by simulating the design being proposed.

ComputerScience

CRC Cards

Order	
Check items are in stock	Order Line
Determine the price	Order Line
Check for valid payment	Customer
Dispatch to delivery address	

ComputerScience

Rules and Practices of XP

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

ComputerScience

Rules and Practices of XP

- 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.

ComputerScience

Acceptance and Unit Tests

- Create tests first - before the code.
- Use the user stories to drive the creation of acceptance tests
- Create or use an automated testing framework
- If you find a bug first create a test to stop it coming back again

ComputerScience

XP Map

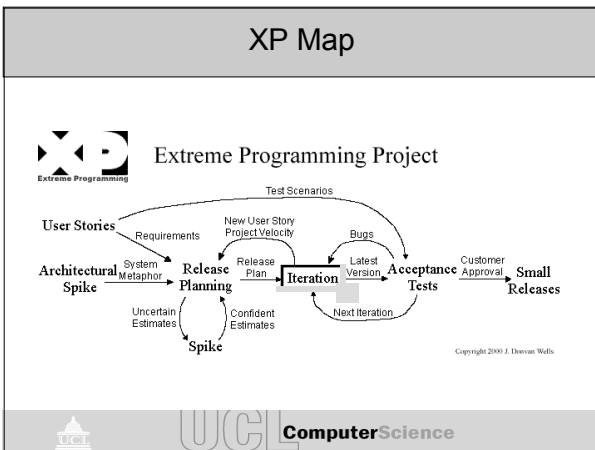Extreme Programming Project

Test Scenarios

User Stories

Requirements

New User Story
Project Velocity

Bugs

Architectural
Spike

System
Metaphor

Release
Planning

Release
Plan

Iteration

Latest
Version

Acceptance
Tests

Customer
Approval

Small
Releases

Uncertain
Estimates

Confident
Estimates

Next Iteration

Spike

Copyright 2000 J. Donvan Wells.

UCL **Computer**Science

Other XP Practices

- Rate user stories by risk - do the hard things first!
- Build the model first with a spartan user interface

Add your own practices
XP is not fixed!

UCL **Computer**Science

Key Points

- There are alternatives to standard OO development processes
- XP works very well in certain situations but IS NOT an excuse for lack of development discipline, quite the contrary
- The core of XP is delivering exactly what is wanted now and regularly refactoring. These practices can be applied in other settings.

UCL **Computer**Science