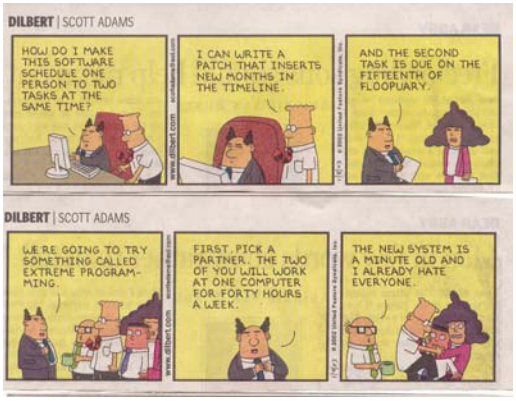
An Introduction to

eXtreme Programming

Jonathan I. Maletic, Ph.D.
<SDML>
Department of Computer Science
Kent State University

Introduction

- Extreme Programming (XP) is a (very) lightweight incremental software development process.
- It involves a high-degree of discipline from the development team
- Popularized by K. Beck (late 90's)
- Comprised of 12 core practices
- Most novel aspect of XP (as a process) is the use of pair programming



Motivational Principles

- Rapid feedback – from customer
- Assume simplicity – keep designs simple
- Incremental change – small changes keep things manageable
- Embracing change – keep your options open
- Quality work – strive for high quality products

XP Core Practices

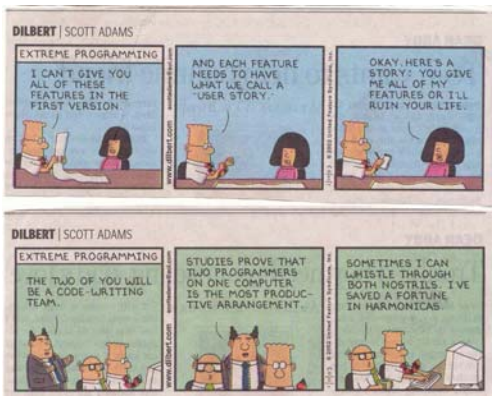
- Planning
- Small Releases
- System Metaphor
- Simple Design
- Continuous Testing
- Refactoring
- Pair Programming
- Collective Code Ownership
- Continuous Integration
- 40-Hour Work Week
- On-site Customer
- Coding Standards

The Planning Game

- Business (customers) and development (programmers) cooperate to produce the maximum business value as rapidly as possible.
- The planning game happens at various scales, but the basic rules are pretty much the same.

Planning Rules

- Business comes up with a list of desired features for the system. Each feature is written out as a *User Story*, which gives the feature a name, and describes, broadly, what is required.
- Development estimates how much effort each story will take, and how much effort the team can produce in a given time interval (an *iteration*).
- Business then decides which stories to implement in what order, as well as when and how often to produce a production releases of the system.



Small Releases

- Start with the smallest useful feature set.
- Release early and often, adding a few features each time.
- Each iteration ends in a release.

System Metaphor

- Each project has an organizing metaphor, which provides an easy to remember naming convention.
- The names should be derived from the vocabulary of the problem and solution domains

Simple Design

- Always use the simplest possible design that gets the job done.
- The requirements will change tomorrow, so only do what's needed to meet today's requirements.
- Uses the fewest number of classes and methods

Continuous Testing

- Before programmers add a feature, they write a test for it. When the suite runs, the job is done.
- Tests in XP come in two basic flavors.
  - Unit Tests
  - Acceptance Tests

Unit Testing

- Unit Tests are automated tests written by the developers to test functionality as they write it.
- Each unit test typically tests only a single class, or a small cluster of classes.
- Unit tests are typically written using a unit testing framework (e.g., JUNIT, ParaSoft).

Acceptance Testing

- Acceptance Tests (Functional Tests) are specified by the customer to test that the overall system is functioning as specified. They typically test the entire system, or some large part.
- When all the acceptance tests pass for a given user story, that story is considered complete.
- At the very least, an acceptance test could consist of a script of user interface actions and expected results that a human can run.
- Ideally acceptance tests should be automated, either using a unit testing framework, or a separate acceptance testing framework.

Refactoring

- Refactor out any duplicate code generated in a coding session.
- You can do this with confidence that you didn't break anything because you have the tests.
- *Refactoring- Improving the Design of Existing Code*, by M. Fowler, 1999 Addison-Wesley

Example of Refactoring

Remove Assignments to Parameters

```
int discount (int inputVal, int quantity, int yearToDate)
{
    if (inputVal > 50) inputVal -= 2;
    ...

int discount (int inputVal, int quantity, int yearToDate)
{
    int result = inputVal;
    if (inputVal > 50) result -= 2;
    ...
```

Pair Programming

- All production code is written by two programmers sitting at one machine.
- Essentially, all code is reviewed as it is written.
- Helm – keyboard and mouse doing implementation
- Tactician – Thinking about the implications and possible problems

Experiences using Pair Programming

- Reported productivity person month [R. Jensen]
  - Single programmer 77 source lines (historical base line)
  - Pair programming 175 source lines
- Cockburn & Williams –
  - Development costs are an additional 15%
  - Resulting code has about 15% fewer defects

Collective Code Ownership

- No single person "owns" a module.
- Any developer is expect to be able to work on any part of the code base at any time.
- Improvement of existing code can happen at anytime by any pair

Continuous Integration

- All changes are integrated into the code base at least daily.
- The tests have to run 100% both before and after integration.

40-Hour Work Week

- Programmers go home on time. In crunch mode, up to one week of overtime is allowed.
- Multiple consecutive weeks of overtime are treated as a sign that something is very wrong with the process.

On-site Customer

- Development team has continuous access to a real live customer, that is, someone who will actually be using the system.
- For commercial software with lots of customers, a customer proxy (usually the product manager) is used instead.

Coding Standards

- Everyone codes to the same standards.
- Ideally, you shouldn't be able to tell by looking at it who on the team has touched a specific piece of code.

Scalability (Team Size)

- XP works well with teams up to 12-15 developers.
- It tends to degrade with teams sizes past 20
- Work has been done in splitting large projects/teams into smaller groups and applying XP within each group.

Environment

- Programmers must be located physically close, often in the same room and desk.
- Iterations typically last 1-3 weeks. Teams will typically use the same duration for all iterations.
- Tests are written before the code is written.
- End of iteration delivers a working system

References and Resources

- *Extreme Programming Explained*, Kent Beck, 2000, Addison Wesley
- There are a number of XP books by AWL (e.g., *XP Installed*, *XP Explored*, and *XP Exaggerated*)

- www.jera.com/techinfo/xpfaq.html
- www.extremeprogramming.org/
- www.xprogramming.com/
- www.pairprogramming.com/