# XP

Peter Dolog
dolog [at] cs [dot] aau [dot] dk
5.2.47
Information Systems
February 29, 2008

# Goal

Design

Tutorial on Requirements Eng. and SCRUM reflections (D401b, s601b)

XP

- Principles
- Process
- Techniques

# eXtreme? => key principles

If code reviews are good, we'll review code all the time (pair programming).

If testing is good, everybody will test all the time (unit testing), even the customers (functional testing).

If design is good, we'll make it part of everybody's daily business (refactoring).

If simplicity is good, we'll always leave the system with the simplest design that supports its current functionality (the simplest thing that could possibly work).

Source: Kent Beck XP Explained

# eXtreme?

If architecture is important, everybody will work defining and refining the architecture all the time (metaphor).

If integration testing is important, then we'll integrate and test several times a day (continuous integration).

If short iterations are good, we'll make the iterations really, really short - seconds and minutes and hours, not weeks and months and years (the Planning Game).

Source: Kent Beck XP Explained

# XP's solutions

*Schedule slips*

Short cycles and frequent releases.

*Project canceled*

The Customer chooses the smallest release that give maximal value. So less could go wrong and the value of the software is greatest.

*System goes sour (too many errors)*

A comprehensive suite of tests are run after every change. The system is kept in prime condition.

*Defect rate*

Tests from perspective of both programmers and customer.

Source: Kent Beck XP Explained

# XP's solution II

*Business misunderstood*

The customer is an integral part of the team
The specifications are continuously refined.

*Business changes*

Shorter release cycles
During a release the customer could substitute or provide new functionality

*False features*

Only the highest priority tasks are addressed

*Staff turnover*

Programmers accept responsibility for estimating and completing their own work
Human contacts are encouraged among the team
Team members are treated as intelligent species

Source: Kent Beck XP Explained

# Four constraints in priority order

The system (code and tests together) must communicate everything you want to communicate.

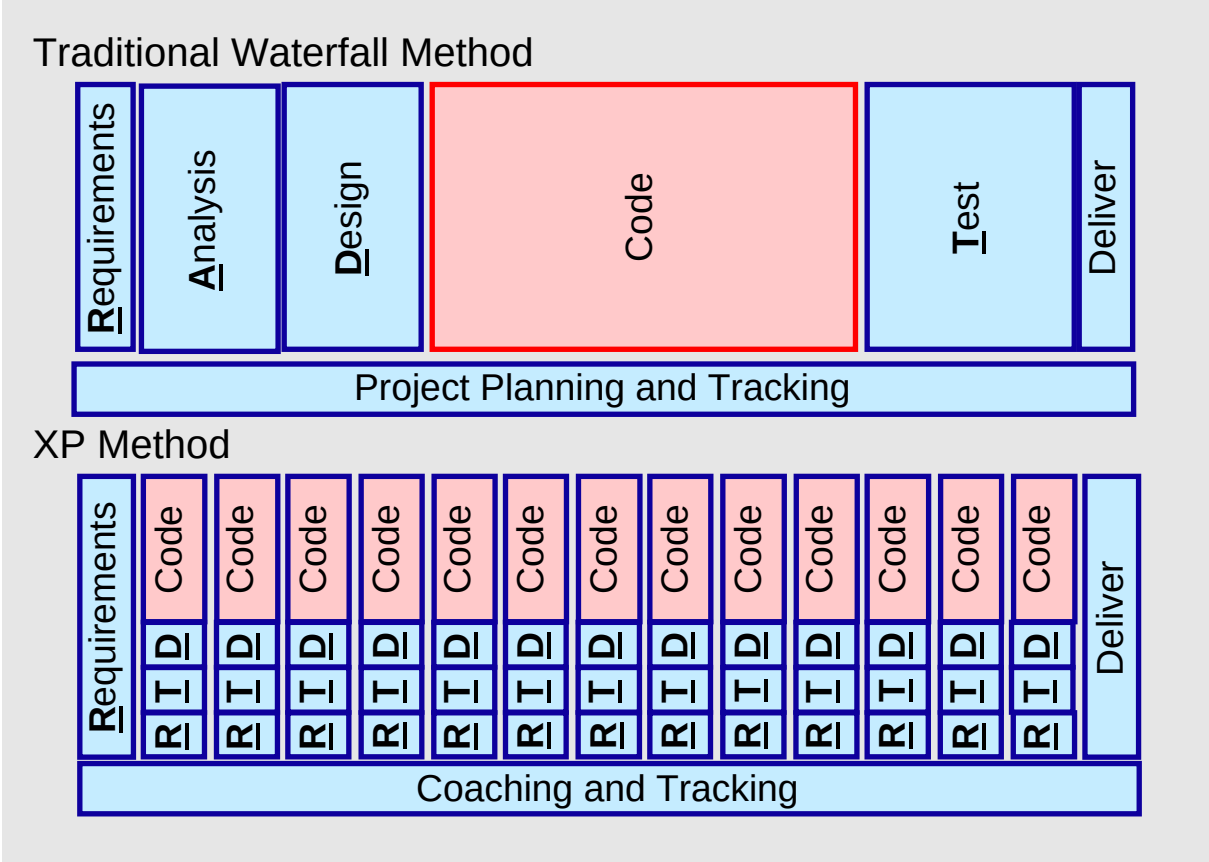
The system must contain no duplicate code. (1 and 2 together constitute the Once and Only Once rule).

The system should have the fewest possible classes.

The system should have the fewest possible methods

Source: Kent Beck XP Explained

# Load Factor Less Than 100%



# XP Planning

## Release Plan

| Release Point 1 | | | | Release Point 2 | | | | Release Point 3 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 ----- | 5 ----- | 9 ----- | 13 ----- | 17 ----- | 21 ----- | 25 ----- | 29 ----- | 33 ----- | 37 ----- | 41 ----- | 45 ----- |
| 2 ----- | 6 ----- | 10 ----- | 14 ----- | 18 ----- | 22 ----- | 26 ----- | 30 ----- | 34 ----- | 38 ----- | 42 ----- | 46 ----- |
| 3 ----- | 7 ----- | 11 ----- | 15 ----- | 19 ----- | 23 ----- | 27 ----- | 31 ----- | 35 ----- | 39 ----- | 43 ----- | 47 ----- |
| 4 ----- | 8 ----- | 12 ----- | 16 ----- | 20 ----- | 24 ----- | 28 ----- | 32 ----- | 36 ----- | 40 ----- | 44 ----- | 48 ----- |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |

Features

Iterations

## Iteration Plan

**Feature 13**
**task**
**task**
**task**
**Feature 14**
**task**
**task**
**task**
**Feature 15**
**task**
**task**
**task**
**Feature 16**
**task**
**task**
**task**

Release Plan contains ...

- **Small Releases** (1-3 months apart)
- Equal size iterations (2 weeks)
- Features (no detail developer tasks)
- Feature effort estimates

Iteration plan contains…

- Features
- Detail Developer tasks (less than 2 days)

# The Planning Game

Business writes a story describing desired functionality
Stories are written on index cards
Development estimates stories
Velocity determines number of stories per iteration
Business splits and prioritizes stories and determines the composition of releases
Velocity is measured and adjusted every iteration
Customer steers development

# XP Planning

**Release Planning** On-site customer and development team collaborate to:

- Identify software features
- Document the features with "stories"
- Prioritize the features
- Estimate the features
- Schedule features into iterations

**Iteration Planning**

In 2 week cycles, the development team will:

- Define detailed tasks for each feature
- Sign up for tasks and re-estimate at the task level
- Test, Design, Code, repeat

Iteration scope and resources are flexible, but not time or quality

# XP Planning Jargon

Features are estimated in Work Units. A work unit is pure coding time, not including:

- Design time
- Testing time
- "Spikes" (experiments)
- "Sand" (small bugs)
- Refactoring
- Non Programming tasks

Velocity is estimated in Work Units per Iteration.

Load Factor is Velocity per Available Programmer Days.

# Stand up meetings

Stand up meetings occur every morning at a scheduled time
VERY short meetings ( goal 5 minutes or less )
Each team member briefly describes:

- Task status
- Issues
- Lessons learned
- Plan for the day

Task completion is recorded - Successes are celebrated
Software engineering principles and suggestions to improve velocity and methodology are briefly discussed
Iteration metrics are updated and posted near the task board - Project status is clearly visible

# Stand up meetings

# User Stories

User stories = lightweight use cases
2-3 sentences on a file card that:

- The customer cares about
- Can be reasonably tested
- Can be estimated & prioritized

# Sample Stories (from XP Installed)

*Union dues vary by union, and are taken only in the first pay period of the month. The system computes the deduction automatically. The amount is shown in the attached table.*

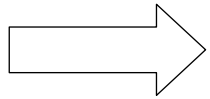
# Splitting a large story

*Allow the user to add new service types to the system's initial list. For example, he may wish to add a special entry for getting the car washed at the high school's "free" wash. Include the standard fields amount and date, plus allow the user to add any additional text or numeric fields. Reports should automatically sum any numeric fields. (Programmer note: story needs splitting. Please separate text and numeric fields into two stories, plus one for the summing.)*

*(Split 1) Allow the user to add new service types, including the standard fields plus any additional text fields desired.*

*(Split 2) Allow the user to add numeric fields to user defined service types.*

*(Split 3) In all reports, show totals of all numeric fields, not just the standard gallons and dollar amount fields.*



**Sketches**

**CRC Cards**

**Quality criteria**

# Promises To Programmers

They will be able to work on things that really matter, every day.

They won't have to face scary situations alone.

They will be able do everything in their power to make their system successful.

They will make decisions that they can make best, and they won't make decisions they aren't best qualified to make.

# Promises to customers & managers

They will get the most possible value out of every programming week.

Every few weeks they will be able to see concrete progress on goals they care about.

They will be able to change the direction of the project in the middle of development without incurring exorbitant costs.

# System Control Variables

Cost

- Often the most constrained variable
- Throwing more money at a problem does not always solve it

Time

- More time can improve quality and increase scope

Quality

- The most difficult control variable (not as easy to measure)
- External quality and internal quality

Scope

# Variables Control

External forces (managers, customers) get to pick the value of 3 of the 4 variables

Development team picks the value of the 4th

Cost?

Time?

Quality?

Scope?

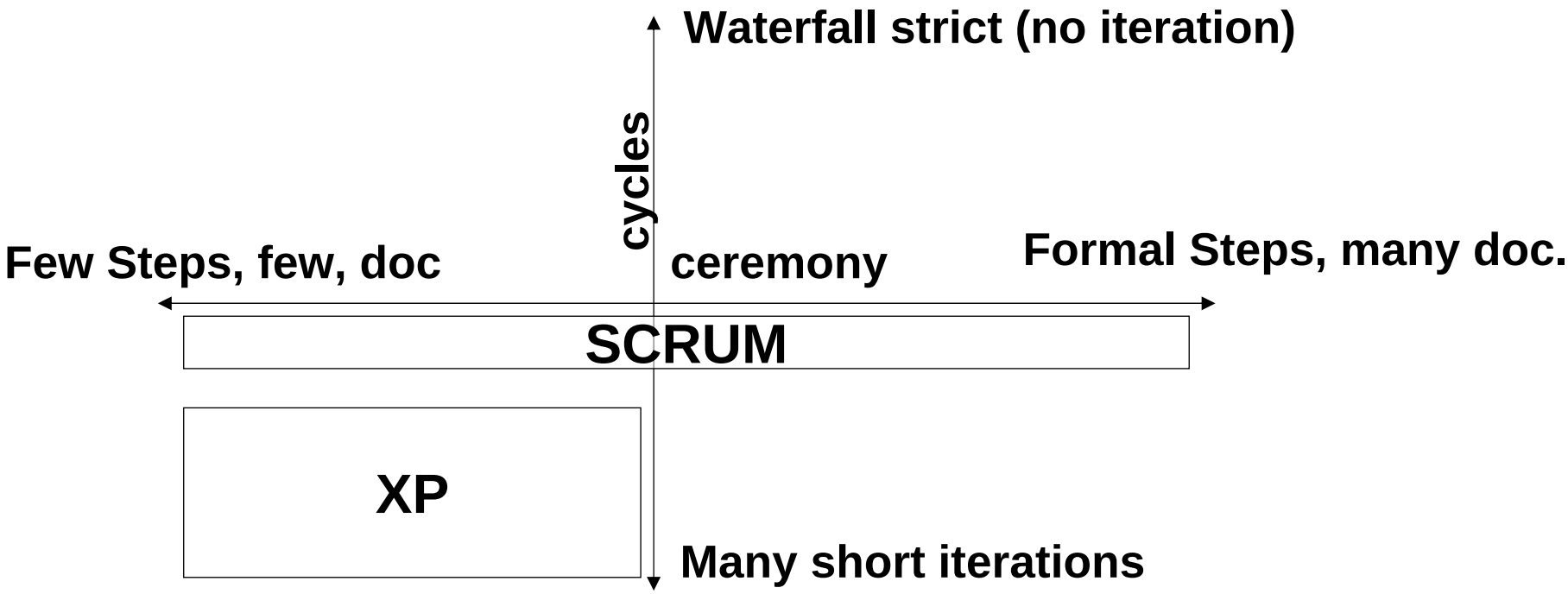
# Cockburn scale

**Criticality**

| | | | | |
|---|---|---|---|---|
| Life (L) | L6 | L20 | L40 | L100 |
| Essential Money (E) | E6 | E20 | E40 | E100 |
| Discretionary Money (D) | D6 | D20 | D40 | D100 |
| Comfort (C) | C6 | C20 | C40 | C100 |

**People**

# Degree of Ceremony and Cycles



# An Attempt to Compare Paradigms

| | Traditional | Agile |
|---|---|---|
| Why | Method det. | Product vision |
| What | Model det. | Code |
| When | Sequential | Iterative |
| Who | Roles | Team |
| Where | Diff. places | Same place |
| How | Description | Acting |
| How much | Method det. | Cust. dec. |