Software Engineering in an On Demand World

ibm.com's experience with eXtreme Programming

David Leip

STSM, ibm.com CTO, and Agile Methods Advocate

Extreme Programming (XP)

"XP is a light weight methodology for small-to-medium-sized teams developing software in the face of vague or rapidly changing requirement."

-- Kent Beck

'eXtreme Programming eXplained'

Back-up Charts

Most Software Projects

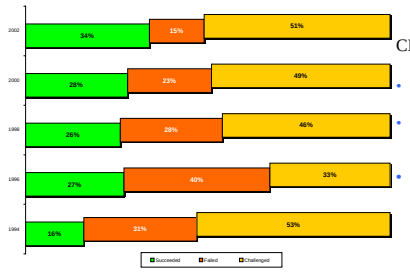
- Late
- Over Budget
- Buggy
- Miss the mark

The Unquestioned Truth: Up-front requirements analysis, design, and modeling are the best way to avoid disaster.

But: For most applications, the actual requirements, even perhaps the real problem, are unknown or not well understood in advance.

Back-up Charts

Failure in S/W Engineering Projects.



http://www.standishgroup.com/sample_research/chaos_1994_1.php

CHAOS Report, Standish Group

- 66% of projects failed or are challenged in 2002
- Large projects are failing more often than small projects
- Only 52% of features make it into product

Back-up Charts

Heavyweight Methodologies

- Logical reaction to the state of affairs.
- Goal: define a rigorous, quantifiable development process, and follow it.
- Emphasis on artefacts (diagrams, models, documents) and formal communication.
- Gives managers something concrete to do, control, and believe in.
- Heavyweight, prescriptive, anti-creative, high overhead, often hated by those who have to use it.

Back-up Charts

What's Needed in an onDemand World?

A methodology that easily accommodates changes in direction.

A pay as you go methodology.

A methodology that allows the customer to decide what is delivered next, function by function.

A methodology that delivers visible results frequently.

A methodology that keep the cost of change low.

Back-up Charts

Enter…

eXtreme

Programming

"The Mountain Dew-fuelled all-nighter is history. Today's supercoders work 40 hours a week. And two to a computer. It's called extreme programming - and it's revolutionizing the software world." –Wired Magazine

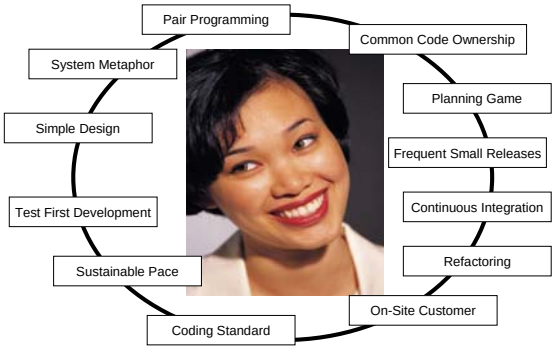
Summary: Agile Software Processes

- Increase responsiveness of software teams
  - Changing requirements
  - Strong customer involvement
- Focus on people, collaboration, communication
  - Focus on face-to-face communication rather than documentation
  - Generalists versus role specialization
  - Plan and correct
  - Customer-focused
- Each release delivers potentially shippable or deployable functionality
- Test focused
- Time-boxed iterations (2 weeks – 4 months)
- Fast feedback

eXtreme Programming Values

- Communication
- Simplicity
- Feedback
- Courage

Extreme Programming Practices



Cost of Pair Programming

Williams, Laurie, Kessler, Robert R., Cunningham, Ward, and Jeffries, Ron, Strengthening the Case for Pair-Programming, IEEE Software, July/Aug 2000

- University study with 41 students
- Higher quality code
  - Test cases passed individuals: 73.4%-78.1%
  - Test cases passed pairs: 86.4%-94.4%
- Pairs completed assignments 40-50% faster (average 15% higher costs)
- Pair programming preferred by students (85%)

User Stories

- Traditionally written on 5x7 index cards
- Describes the interaction with the system from the user's perspective.
  > "The user presses the New Instrument button, selects bond instruments, then enters the bond details into the system. On completion, he presses Accept."
- Associate a title and contact customer for each story.

Selecting User Stories

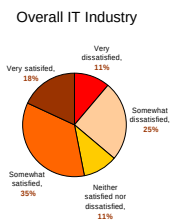
- Selecting stories is like shopping
  - The items (stories)
  - The prices (time estimates)
  - The budget (the allotted time & manpower)
  - The constraints (business and technology constraints)
- Customers pick the items whose prices fit into the budget and satisfy the constraints.
- Order stories by business value
- High risk stories may be done earlier
- Slot stories into iterations using velocity

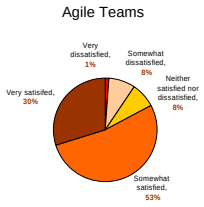
Yesterday's Weather (Velocity)

- How do you estimate how long it will take to implement stories?
- Measure how many features you implemented in the last iteration
- Estimate that the next iteration will implement the same number of features

Job Satisfaction Study (Melnik/Maurer)

Do agile methods lead to higher job satisfaction rates in software development teams than the average satisfaction in IT industry?





Skill Building

- Created Study Group
- Brought in experts to help
- Extreme Construction Session
  - http://csis.pace.edu/~bergin/extremeconstruction/
- XP text books for all on the team.
- Third party coaches.

Some things we learned

- Don't confuse simple philosophy with being simple to do.
- Training is important.
- Get strong coaches.
- Devote someone to tools, if you can.
- Nomenclature can be important for buy-in.
  - "Planning Game" for example can create negative perceptions for example with executives.
- Automated test cases for much of the UI can be fragile, and thus of little value.

Some more things we learned

- System metaphor concept didn't work for us.
- Take care to keep concept of ideal time and real time clearly separated.
- You might need to segment the team into different skill areas.
- You don't need to pair for everything.
- Stand-up meetings can degenerate into customer status meetings if you are not careful.
- Be ready to adapt. Stop doing things that are not adding value. But be careful that you reinforce what you stop doing in other ways.

Results & Further Work/Research

- Corporate Portal 4.0 successfully deployed on schedule (Nov. 2004)
  - Satisfied Customer
- Preferred Tools
- Further integration with IT governance systems.
- Meshing s/w development velocity with other velocities.
- XP and Usability.

Further Information

- David Leip Leip@us.ibm.com
- http://www.ExtremeProgramming.org

Back-up Charts!

Back-up Chart Index

- XP Bill of Rights
- XP Roles
- Stages of an XP Project
- Communications

XP Customer Bill of Rights

- As the customer, you have the right to:
  - An overall plan, to know what can be accomplished, when, and at what cost;
  - Get the most possible value out of every programming week;
  - See progress in a running system, proven to work by passing repeatable tests that you specify;
  - Change your mind, to substitute functionality, and to change priorities without paying exorbitant costs;
  - Be informed of schedule changes, in time to choose how to reduce scope to restore the original date, even cancel at any time and be left with a useful working system reflecting investment to date.

XP Developer Bill of Rights

- As the Developer, you have the right to:
  - Know what is needed, with clear declarations of priority;
  - Produce quality work at all times;
  - Ask for and receive help from peers, superiors, and customers;
  - Make and update your own estimates;
  - Accept your responsibilities instead of having them assigned to you.

XP Roles

- Customer
  - Writes User Stories and specifies Functional Tests
  - Sets priorities, explains stories
  - May or may not be an end-user
  - Has authority to decide questions about the stories
- Programmer
  - Estimates stories
  - Defines Tasks from stories, and estimates
  - Implements Stories and Unit Tests
- Coach
  - Watches everything, sends obscure signals, makes sure the project stays on course
  - Helps with anything
  - Applies "Rolled Up Newspaper" as required

XP Roles *(cont.)*

- Tracker
  - Monitors Programmers' progress, takes action if things seem to be going off track.
  - Actions include setting up a meeting with Customer, asking Coach or another Programmer to help
- Tester
  - Implements and runs Functional Tests (not Unit Tests!)
  - Graphs results, and makes sure people know when test results decline.
- Doomsayer
  - Ensures that everybody knows the risks involved
  - Ensures that bad news isn't hidden, glossed over, or blown out of proportion

Stages of an XP Project

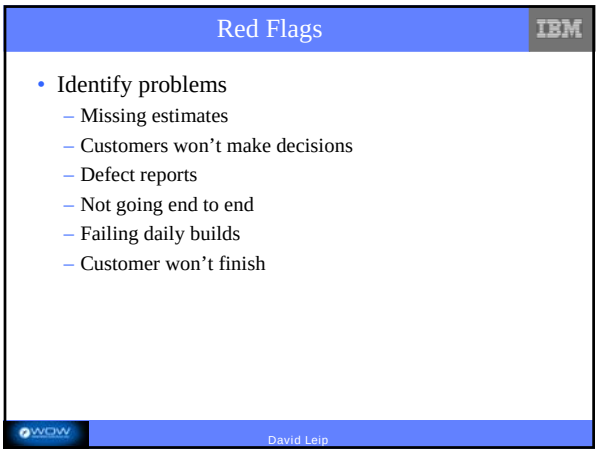
- Initiation
  - User Stories
  - Release Planning
  - Release (typically 1-6 months)
    - Iteration 1 (typically 1-3 weeks)
      - Development
      - Deployment
      - Acceptance Testing
    - Iteration 2
      - Development
      - Deployment
      - Acceptance Testing
    - …
    - Iteration n

Stand-up Meetings

- Daily meetings
- Everyone has to stand for the whole meeting
- What did you do yesterday?
- What are you doing today?
- Problems or announcements of interest to the team are raised
- Don't try to solve problems – take it offline

Visible Graphs

- Smell a problem
- Devise a measurement
- Display the measurement
- If the problem doesn't go away, repeat
- Choose graphs carefully. Use it only as long as needed.


Red Flags
IBM
• Identify problems
– Missing estimates
– Customers won’t make decisions
– Defect reports
– Not going end to end
– Failing daily builds
– Customer won’t finish
David Leip