

CS 350, slide set 14

---

M. Overstreet  
Old Dominion University  
Spring 2005

---

---

---

---

---

---

---

---

Final Exam

---

- Comprehensive
- Take-home part & in-class part
- Final is 3:45 – 6:45, Wednesday, May 4
- Take-home due (hardcopy) at start of in-class final

---

---

---

---

---

---

---

---

Refs

---

- 1. <http://www.agilealliance.org>
  - Agile alliance manifesto:
    - Individuals and interactions over processes and tools
    - Working software over comprehensive documentation
    - Customer collaboration over contract negotiation
    - Responding to change over following a plan
  - XP series of books from Addison-Wesley
    - McBreen: Questioning Extreme Programming

---

---

---

---

---

---

---

---

## Topic: Views on Extreme Programming (XP)

- Loved by programmers
- Factor 6 improvement in productivity!
- No documentation needed: code is documentation
  - If not, bad code. Recode.
- Glorifies "cowboy coders"
- Hackers paradise
- Sets software engineering back 30 years!
- Lots of flame wars!

---

---

---

---

---

---

---

---

## Agile Programming

- Small agile team – 5 to 10
  - Dominated by people with extensive project experience
  - Can only tolerate a few inexperienced enthusiastic beginners
- Regular delivery schedule – incremental system delivered to customer every few weeks
- Customer co-located with development team
- Customer decides what to include in next increment
  - "Must have," "should have," "can have"
  - With advice from team (feasibility, difficulty)

---

---

---

---

---

---

---

---

## XP & CMM Characterizations - 1

- In 60's, big software systems were replacing existing manual systems
  - What was needed was pretty well understood
- Today's systems (web apps, e.g.) are fundamentally different.
  - Requirements cannot be known at start

---

---

---

---

---

---

---

---

## CMM & XP characterizations – 2 (as seen by XP advocates)

- CMM designed by managers
  - Addresses management needs
  - Traditional SE processes minimize risk of making mistakes even at cost of making proceed at glacial pace
- CMM is based on fear:
  - Don't let what happened on last project happen on this one
  - Need reviews, approvals, committees, forms to avoid problems
- XP designed by programmers
  - Based on Smalltalk development systems
    - Programming env. developed at Xerox Parc
    - Apple stole interface from Xerox for Lisa (before Mac)
    - Windows stole interface from Apple. Apple sued. & lost
  - Addresses programmer needs

---

---

---

---

---

---

---

---

## How to decide what's better?

- Fact: Hawthorne Effect (Western Elec. plant in Chicago, 1930's)
  - Beginning of industrial engineering
  - Goal: improve worker productivity
    - Increase light levels: prod. up
    - Give rest breaks: prod. up
    - Decrease light levers: prod. up
    - Decrease again: prod. up
    - Put light levels back to original: prod. up
- What's going on?
  - Expectations effect results?
  - Excitement of participating in a study?
  - Knowing people are watching?
- Makes medical experimentation more difficult!

---

---

---

---

---

---

---

---

## Perceived problem: change expensive, get it right the first time

- Do exhaustive analysis; build system so that **all** foreseeable **changes** made with configuration changes
  - Smalltalk problem: programmers tended to build a wonderful environment for developing a product rather than developing a product.
- Do analysis; build system so that **likely changes** can be made easily
- Write components so that **dependencies** are **minimized**; easy to change parts as necessary
- Focus on getting current version **shipped**; changes, if needed, will come from another budget

---

---

---

---

---

---

---

---

## XP influenced by programmer worries:

- Schedule slippages
- Cancelled project
- System goes sour
- Defect rate
- Problem misunderstood
- World changes
- Feature-rich software
- Staff turnover
- If you've been part of a failed project, damages your career for a long time

---

---

---

---

---

---

---

---

## XL "solutions" - 1

- Schedule slippages
  - Deliver new version every few weeks
- Cancelled project
  - Get product in hands of users
- System goes sour
  - Tight delivery schedule
  - Tight coupling with customer on site
  - Feedback from users

---

---

---

---

---

---

---

---

## XP "solutions" - 2

- Defect rate
  - Extreme unit testing
- Problem misunderstood
  - Customer co-located with development team
- World changes
  - Customer co-located with development team

---

---

---

---

---

---

---

---

## XP "solutions" - 3

---

- Feature rich software  
(Fun for inexperienced programmers; feared by experienced programmers)
  - Customer selects features based on need
- Staff turnover  
(More concern about other people leaving)
  - People stick with exciting successful projects
    - They're challenging and fun

---

---

---

---

---

---

---

---

## XP life cycle - 1

---

- **Exploration** - short period. team experiments with technology to gain confidence in estimating what's needed to satisfy customer desires
- **Planning** - very short; team creates initial release plan for the date at which the smallest, most valuable capabilities will be done

---

---

---

---

---

---

---

---

## XP life cycle - 2

---

- **Iteration to First Release** - team works in short iterations to delivery tested functionality. Every 3 or 4 releases, release plan (step 2) is updated to reflect team performance and changed requirements
- **Productizing** – End game of initial iteration, leads to 1<sup>st</sup> release of production version

---

---

---

---

---

---

---

---

## XP life cycle - 3

- **Maintenance** – normal state of an XP product. regular iterations resulting in new and changed features for production use
- **Death of project** – customer cannot think of any more features (or unwilling to pay for more). team writes documentation needed for future enhancements then is disbanded

---

---

---

---

---

---

---

---

## What's XP missing?

- No efficiency gains through specialization
  - Part of the fun of XP is that every team member designs, codes, tests
- No detailed project plans
  - How do we know when we're 50% done?
- No comprehensive documentation to ensure auditability and traceability
- (Almost) no second set of eyes checking people's work
  - Though XP uses "Pair Programming:" people work in pairs

---

---

---

---

---

---

---

---

## Other development methodologies

- Game development
  - Driven by predictable delivery of a stable product
  - Why? Marketing budgets
  - See <http://www.gamasutra.com>
- Open source (linux primary example)
  - No shortage of developers
  - People develop many competing, duplicate items
  - Management committee then selects the best

---

---

---

---

---

---

---

---

## Is XP the solution?

- Sometimes. Programmers really like it.
- What if you need 200 people?
- What about software to fly new Airbus or Boeing aircraft?
  - Requirements relatively well understood
- Many organizations will not put customer on site with developers, too expensive
- More?

---

---

---

---

---

---

---

---