CSCI 335 — Software Development

eXtreme Programming

Feb 23, 2009

Material adapted from slides by Jan Vitek at Purdue University and Kent Beck, *eXtreme Programming eXplained*.

Problems in Software Development

Schedule slips. Estimates are hard to make, even harder to meet.

System souring. After a few years in place and a few changes, things stop working.

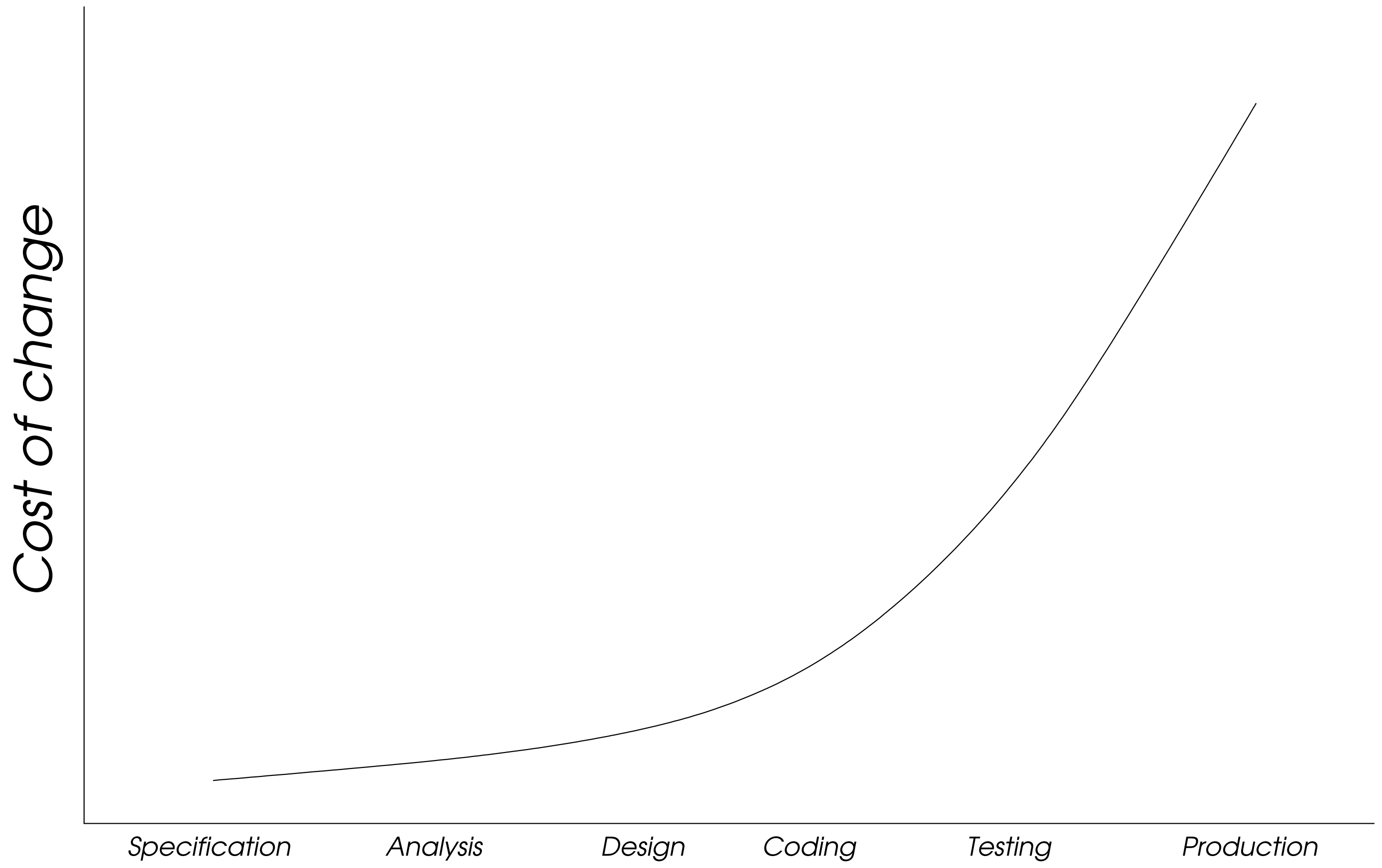
Defect rate. When code is buggy enough, it is abandoned.

Specification misunderstandings and changes. The system solves wrong or out-of-date problems.

False feature richness. The system has lots of features, most unnecessary.

Staff turnover. The best programmers quit.

The Traditional View



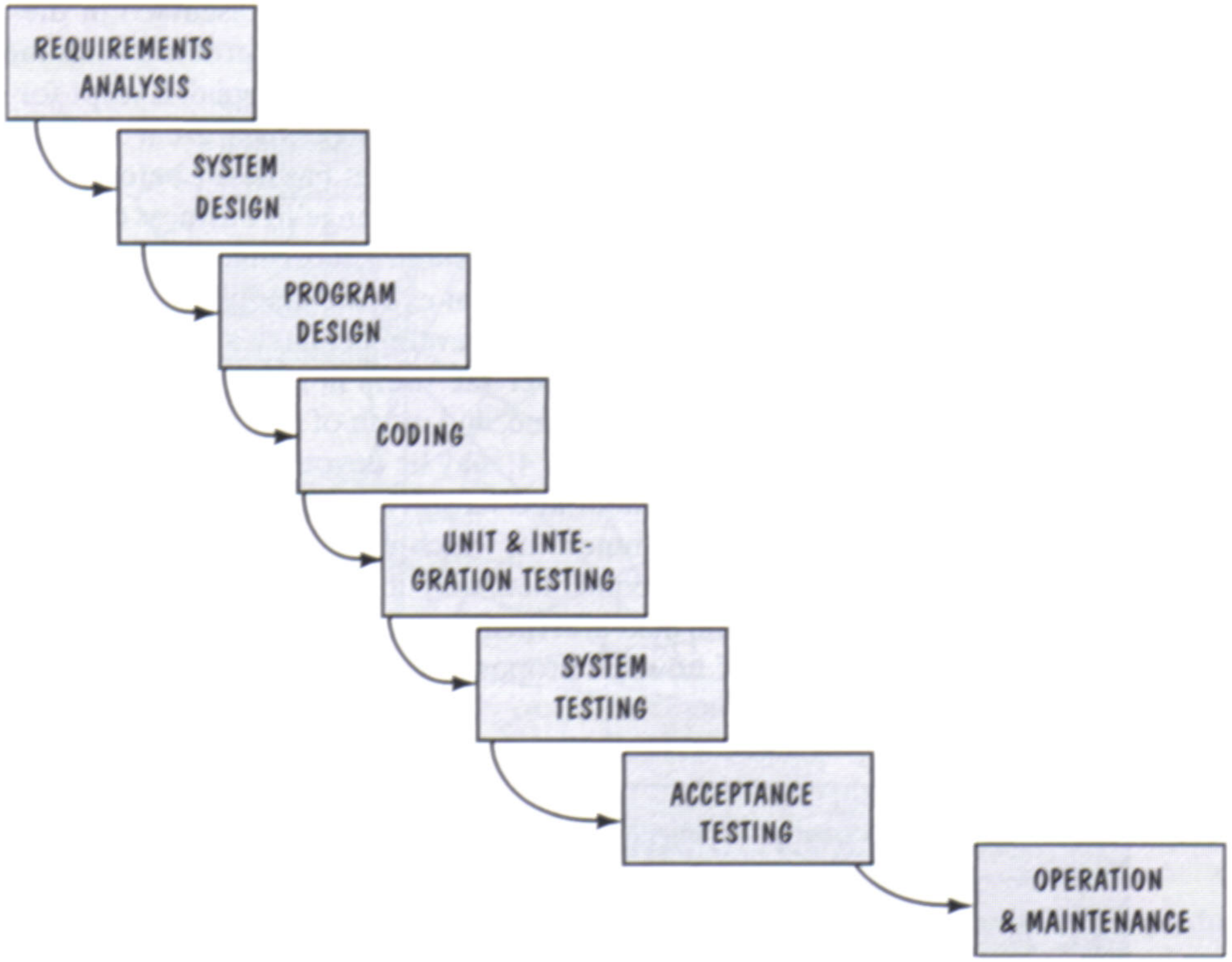


FIGURE 2.1 The waterfall model.

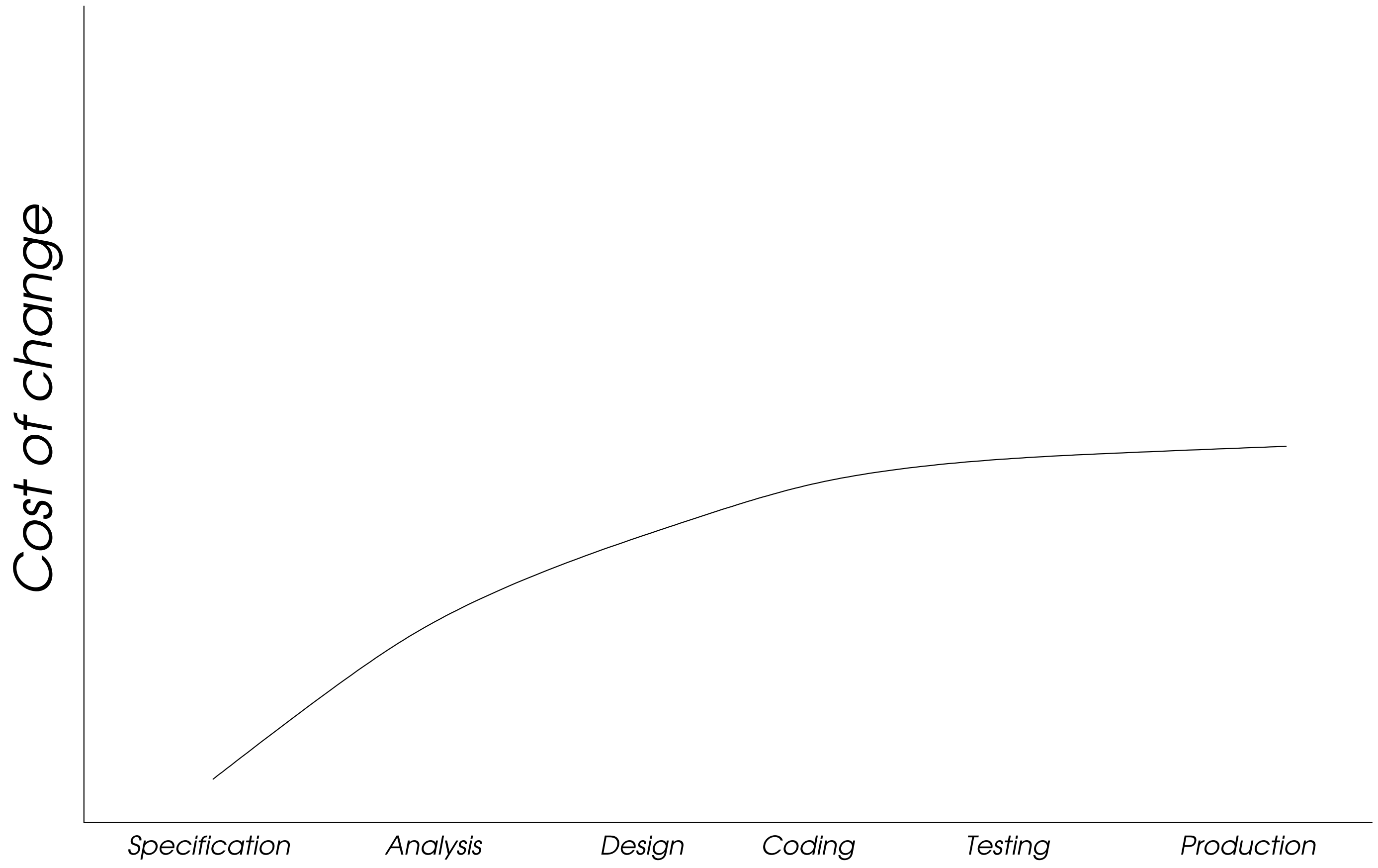
S. L. Pfleeger and J. M. Atlee, *Software Engineering, Theory and Practice*, Prentice Hall, 2006. pg 49.

The Traditional View

Design, design, design ***first!***

We must work out all the bugs ahead of time.
We can't afford mistakes later.

XP's Central Premise



XP's Solutions

- **Schedule slips.**
  - Short release cycle
  - Implement most important features first
- **System souring.**
  - Constant customer involvement
  - Creation and maintenance of test suite
  - Run test after each change
- **Defect rate.**
  - User/customer gives functional tests
  - Programmer makes unit tests

XP's Solutions

- **Spec misunderstanding and changes.**
  - Customer is on the team
  - Change?– No problem
- **False feature richness.**
  - Highest priority features first
- **Staff turnover.**
  - Process breeds loyalty

XP Strategy

Contain the cost of change:

- OO design (for reusability, plus polymorphism etc)
- Simple designs
- Automated tests
- Atmosphere of modifications

Instead of making big decisions early and little ones late, XP makes frequent decisions and assures them with tests.

XP Principles

1. Rapid feedback
2. Assumption of simplicity
3. Incremental change
4. Embracing of change
5. Quality work
6. Small initial investment
7. Concrete experiments
8. Open, honest communication
9. Accepted responsibility
10. Local adaptation
11. Travel light
12. Honest measurements

XP Activities

1. Coding
2. Testing
3. Listening
4. Designing

XP Practices

1. **Whole team.** *Remove the barrier between the customer and the rest of the development team*
2. **Metaphor.** *Design with a shared story.*
3. **The planning game.** *Quick and focused on the next release.*
4. **Small releases.** *Simple system soon, short cycle for new versions.*
5. **Simple design.** *Remove complexity.*
6. **Testing.** *Developers continuously write unit tests, customers write functional tests.*
7. **Refactoring.** *Redesigning while maintaining functionality.*
8. **Pair programming.** *All code produced by two programmers at one machine.*
9. **Collective ownership.** *All code is communal—anyone may change anything at anytime.*
10. **Continuous integration.** *Build many times a day.*
11. **40-hour week.** *No more than two consecutive over-time weeks.*
12. **On-site customer**. *A live user is on the team.*
13. **Coding standards.** *Prevents chaos.*

Critique

- `http://www.globalnerdy.com/2007/11/28/dilbert-on-extreme-and-agile-programming/`
- `http://www.softwarereality.com/lifecycle/xp/safety_net.jsp`
- XP is not for extremely large projects.