



eXtreme Programming

(summary of Kent Beck's XP book)

Slides adapted from

Prof. Dr. Wolfgang Pree
Universität Salzburg

Wolfgang.Pree@cs.uni-salzburg.at

Contents



- The software development problem
- The XP solution
- The JUnit testing framework

The SW development problem

Risk Examples



- **delivery schedule**
- **project cancelled**
- **high defect rate - system unusable**
- **business misunderstood or changed**
- **false feature rich**
- **staff turnover**



Four variables

Overview



- **cost**
- **time**
- **quality**
- **scope**

**external forces (customers,
management) pick the values of 3 v.
solution: make the four variables visible**

interaction between the variables



- time: more time can improve quality and increase scope
too much time will hurt it
- quality: short-term gains by deliberately sacrificing quality; but the cost (human, business, technical) is enormous
- less **scope** => better quality (as long as the business problem is still solved)




Four values

Overview




- **communication**
- **simplicity**
- **feedback**
- **courage**

short-term vs. long term thinking (I)



- communication: effect of pair programming, unit testing, task estimation: programmers, customers and managers have to communicate
- simplicity: it is better to do a simple thing today and pay a little more tomorrow to change it if it needs than to do a more complicated thing today that may never be used anyway

short-term vs. long term thinking (II)



- feedback: when customers write new „stories“ (description of features, simplified use cases), the programmers immediately estimate them; customers and testers write functional tests for all the stories
- courage: throwing parts of the code away and start over on the most promising design



Basic principles (derived from the four values)

Basic principles (I)



- **rapid feedback**
- **assume simplicity**
- **incremental change**
- **embracing change**
- **quality work**

Basic principles (II)



- **small initial investment**
- **play to win**
- **concrete experiments**
- **open, honest communication**
- **work with people's instincts, not against them**



Basic activities

Basic activities in the XP development process



- coding
- testing
- listening
- designing

The solution



XP practices

Practices (I)



- **planning game: determine the scope of the next release; as reality overtakes the plan update the plan**
- **small releases: release new versions on a very short cycle after putting a simple system into production quickly**
- **metaphor: guide development with a simple shared story of how the whole system works**

Practices (II)



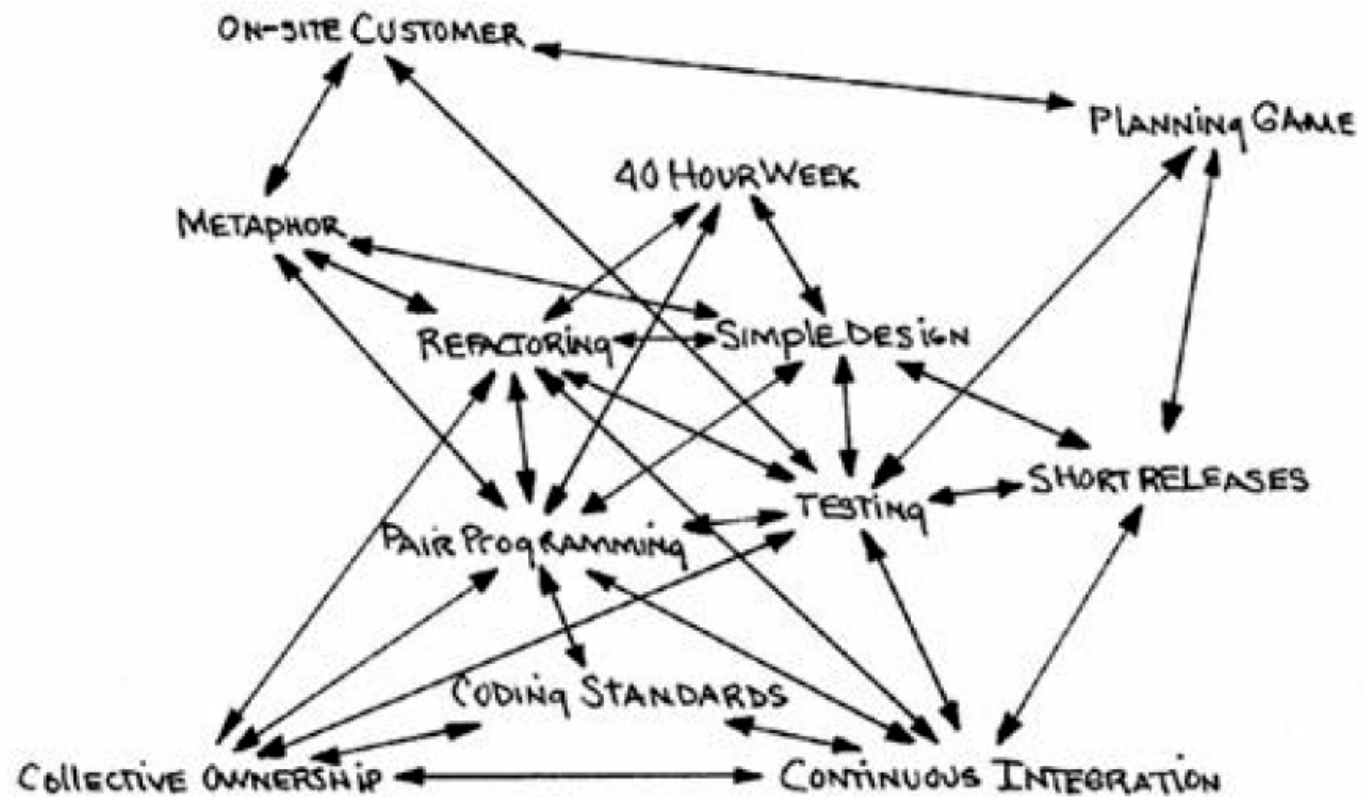
- **simple design: as simple as possible but not simpler (A. Einstein)**
- **testing: continually write unit tests**
- **refactoring: restructure the system to remove duplication (c.f. framelets, etc.)**
- **pair programming: two programmers at one machine**
- **collective ownership**

Practices (III)



- **continuous integration: integrate the system many times a day, every time a task is complete**
- **40-hour week**
- **on-site customer: include a real, live customer**
- **coding standards**

Practices support each other





Management strategy

Overview



- **decentralized decision making based on**
 - metrics
 - coaching
 - tracking
 - intervention
- **using business basics: phased delivery, quick and concrete feedback, clear articulation of the business needs, specialists for special tasks**

Metrics



- **don't have too many metrics**
- **numbers are regarded as a way of gently and noncoercively communicating the need for change**
- **ratio between the estimated development time and calendar time is the basic measure for running the Planning Game**

Coaching



- **be available as a development partner**
- **see long-term refactoring goals**
- **explain the process to upper-level management**

=> no lead programmer, system architect, etc.

Intervention



- **when problems cannot be solved by the emergent brilliance of the team, the manager has to step in, make decisions and see the consequences through to the end**
- **sample situations: changing the team's process, personnel changes, quitting a project**



Planning strategy

Overview



- **bring the team together**
- **decide on scope and priorities**
- **estimate cost and schedule**
- **give everyone confidence that the system can be done**
- **provide a benchmark for feedback**

put the most valuable functionality into production asap



Summary

What makes XP hard?



It's hard to ...

- do simple things
- admit you don't know (eg, basics about computer/software science in the context of pair programming)
- to collaborate
- to break down emotional walls

XP & Kent Beck (I)



Kent Beck is afraid of:

- **doing work that doesn't matter**
- **having projects canceled**
- **making business decisions badly**
- **doing work without being proud of it**

XP & Kent Beck (II)



Kent Beck is not afraid of:

- **coding**
- **changing his mind**
- **proceeding without knowing everything about the future**
- **relying on other people**
- **changing the analysis and design of a running system**
- **writing tests**