

# *Extreme Programming*

*Best Practices, Tradeoffs, and Variants*

**Dr. Mark C. Paulk**  
**Carnegie Mellon University**

*July 28, 2009*  
*PMI - IT & Telecom SIG*  
*Scrum Alliance Webinar Series*

## *Recommended XP Books*

### *The basic XP book*

- **Kent Beck, Extreme Programming Explained: Embrace Change, 1999.**
- **Kent Beck and Cynthia Andres, Extreme Programming Explained: Embrace Change, 2nd Edition, 2004.**

### *The anti-XP book (not anti-agile!)*

- **Matt Stephens and Doug Rosenberg, Extreme Programming Refactored: The Case Against XP, 2003.**

## *Other Books on Agile Methods*

### **Viewing agile methods from a risk management perspective**

- **Barry Boehm and Richard Turner, Balancing Agility and Discipline: A Guide for the Perplexed, 2004.**

### **Comparing agile methods**

- **Craig Larman, Agile and Iterative Development: A Manager's Guide, 2004.**

### **Feature Driven Development (FDD)**

- **David J. Anderson, Agile Management for Software Engineering: Applying the Theory of Constraints for Business Results, 2004.**

3 of 35

## *Topics*

- ➔ **An Overview of XP**
  - XP Management Practices**
  - XP Design Practices**
  - XP Code and Test Practices**
  - Summing Up Agile and XP**

4 of 35

## *Extreme Programming Practices*

Planning game

Small releases

Metaphor

Simple design

Testing

- (test-driven development)
- (customer tests)

Refactoring

- (design improvement)

Pair programming

Collective (code) ownership

Continuous integration

40-hour week

- (sustainable pace)

On-site customer

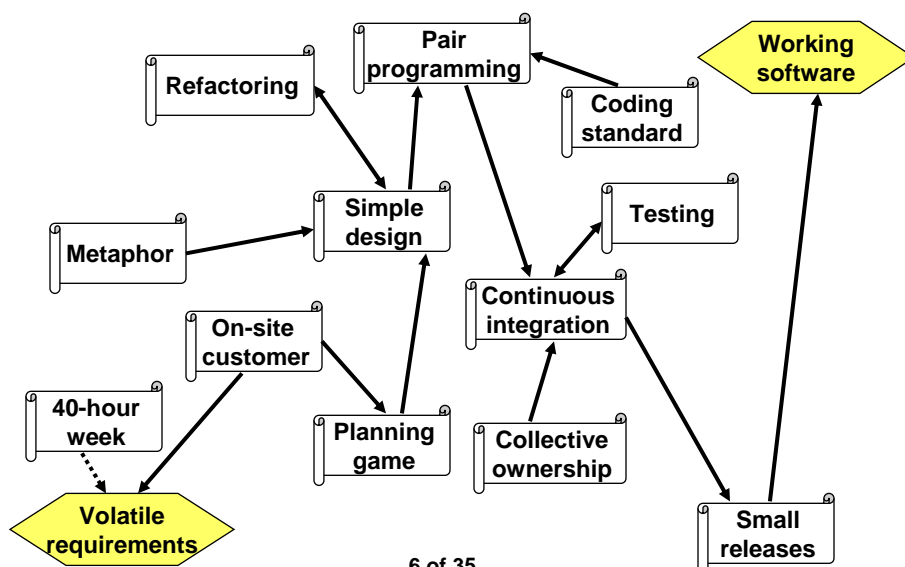
- (whole team)

Coding standard

*Kent Beck, Extreme Programming Explained: Embrace Change, 1999.  
(<http://www.xprogramming.com/xpmag/whatisp.htm>)*

5 of 35

## *(Some) XP Practice Relationships*



6 of 35

## *The 2004 XP Practices*

### **Primary Practices**

- Sit together
- Whole team
- Informative workspace
- Energized work
- Pair programming
- Stories
- Weekly cycle
- Quarterly cycle
- Slack
- Ten-minute build
- Continuous integration
- Test-first programming
- Incremental design

### **Corollary Practices**

- Real customer involvement
- Incremental deployment
- Team continuity
- Shrinking teams
- Root-cause analysis
- Shared code
- Code and tests
- Single code base
- Daily deployment
- Negotiated scope contract
- Pay-per-use

*K. Beck and C. Andres, [Extreme Programming Explained: Embrace Change, 2nd Edition, 2004.](#)*

7 of 35

## *The Sweet Spot for XP*

**Very volatile (emergent) requirements**

**Small to medium sized systems**

**Small teams of highly competent generalists**

**Co-located with empowered customer**

**Not life-critical or essential moneys systems**

8 of 35

## *Topics*

An Overview of XP

→ XP Management Practices

XP Design Practices

XP Code and Test Practices

Summing Up Agile and XP

9 of 35

## *XP: On-Site Customer (Whole Team)*

A real, live user on the team full-time to answer questions.

An on-site customer, or some reasonable proxy, is a prerequisite for the planning game and small releases to work – an implementation prerequisite.

### *Antecedents*

- *Joint Application Design (Davidson 1999)*
- *end-user development (Janvrin 1996; Burnett 2004)*

10 of 35

## *XP: Planning Game*

Quickly determine the scope of the next release, combining business priorities and technical estimates.

The customer decides scope, priority, and dates from a business perspective.

Technical people estimate and track progress.

### *Antecedents*

- *JAD*
- *Quality Factor Development (Zultner 1995)*

11 of 35

## *XP: Small Releases*

Put a simple system into production quickly.

Release new versions on a very short (two-week) cycle.

*“If you can’t plan well, plan often.”*

- *Watts Humphrey*

An implementation decision on the right kind of life cycle.

12 of 35

## *Small Release Antecedents*

### *Iterative / incremental / evolutionary life cycles*

- *Mills 1976*
- *McConnell 1996*
- *Larman 2003*

*evo (Gilb 1988)*

*Prototyping (Baskerville 2006)*

*Rapid Application Development (Beynon-Davies 1999)*

## *XP: 40-Hour Week (Sustainable Pace, Energized Work)*

**Work no more than 40 hours per week as a rule.**

**Never work overtime two weeks in a row.**

***Work only as many hours as you can be productive and only as many hours as you can sustain.***

## *Sustainable Pace Antecedents*

*Peopleware (DeMarco 1999)*

*Multitasking, fragmentation (Constantine 1995)*

*Slack (DeMarco 2001)*

*Critical chain project management (Goldratt 1997)*

*People CMM (Curtis 2001)*

15 of 35

## *2004 XP Management Additions*

### **Sit together**

- develop in an open space big enough for the whole team

### **Informative workspace**

- make your workspace about your work... big, visible charts

### **Weekly cycle**

- plan work a week at a time

### **Quarterly cycle**

- plan work a quarter at a time

### **Slack**

- in any plan, include some minor tasks that can be dropped if you get behind.

16 of 35

## *Topics*

An Overview of XP

XP Management Practices

→ XP Design Practices

XP Code and Test Practices

Summing Up Agile and XP

17 of 35

## *XP: Metaphor*

Guides all development with a simple, shared story of how the whole system works.

### *Antecedents*

- *Unique to XP as far as I know!*
- *Not another name for architecture...*

*“Few practitioners claimed to understand the intent of metaphor as a practice, how to devise and evaluate metaphors, or the relationship of metaphor to design and the other XP practices. As XP evolved metaphor was demoted and is no longer a separate practice. Most agile proponents mention metaphor and acknowledge minor uses. This paper argues that XP was wrong to abandon metaphor and advocates a more systematic discussion of how metaphor informs development.” (West 2005)*

18 of 35

## *XP: Simple Design*

Design as simply as possible at any given moment.

An implementation philosophy on how to design.

### *Antecedents?*

- *Structured programming: information hiding, abstraction, high cohesion, low coupling, ...*
  - *Pressman 2004*
  - *plus Dijkstra, Mills, Constantine, Knuth, etc.*

*“... resolution of these design issues was based on a simple rule: do not implement a solution to a problem you may not have.”*

- *M.C. Paulk, “The ARC Network: A Case History,” IEEE Software, Vol. 2, No. 3, May 1985, pp. 62-69.*

19 of 35

## *Tacit vs Explicit Knowledge*

“The difference between agile methods and the Unified Process is knowledge management – agile is tacit, UP is explicit.”

“80% of software work is no-brain work.”

“60% of change requests for MS-Word are for features that are already there.”

- *Ivar Jacobson, Software Engineering Conference (Russia), 27-28 October 2005.*

20 of 35

## *XP: Refactoring (Design Improvement)*

Restructure the system without changing behavior to remove duplication, improve communication, simplify, or add flexibility.

An implementation philosophy on how to design.

### *Antecedents*

- *No antecedents that I am aware of!*
- *Refactoring: Improving the Design of Existing Code (Fowler 1999)*

## *Topics*

An Overview of XP

XP Management Practices

XP Design Practices

→ XP Code and Test Practices

Summing Up Agile and XP

## *XP: Coding Standards*

Rules emphasizing communication throughout the code.

### *Antecedents*

- *DOD, IEEE, ISO standards...*
- *Programming Proverbs (Ledgard 1975)*
- *Programming Pearls (Bentley 1986/88)*
- *Literate programming (Knuth 1984)*

23 of 35

## *XP: Pair Programming*

All production code is written by two programmers at one machine.

An implementation philosophy – and a culture shift! – on how to continuously do peer reviews.

### *Antecedents*

- *Walkthroughs (Weinberg 1971/98)*
- *Dynamic duos (Constantine 1995)*
- *Peer reviews (Paulk 1995)*
- *Pair Programming Illuminated (Williams 2002)*

24 of 35

## *XP: Collective (Code) Ownership*

**Anyone can improve any code anywhere in the system at any time.**

**An implementation philosophy – and a culture shift! – on how to address problems (and make improvements) that depends on “continuous integration” to work.**

### ***Antecedents***

- ***Unique to XP as far as I know...***
- ***Worker empowerment conceptually...***
- ***Depends on continuous integration as an enabling practice!***

25 of 35

## *XP: Continuous Integration*

**Integrate and build the system many times a day, every time a task is finished.**

**Continual regression testing means no regressions in functionality as a result of changed requirements.**

***Integrate and test changes after no more than a couple of hours.***

### ***Antecedents***

- ***Daily build and smoke test***
  - ***Cusumano 1995***
  - ***McConnell 1996***
- ***Compile and smoke test frequently***
  - ***SPMN critical software practice, Lochner 1999***

26 of 35

## *XP: Testing* (*Test-Driven Development; Test-First Programming; Customer Tests*)

**Continually write unit tests that must run flawlessly.**

**Customers write tests to demonstrate functions are finished.**

**“Test, then code” means a failed test case is an entry criterion for writing code.**

### ***Antecedents***

- ***Early reference in NASA’s Project Mercury***
  - *Williams 2003*
  - *Larman 2003*

27 of 35

## *2004 XP Design & Code Additions*

### **Incremental design**

- **invest in the design of the system every day**
- **the question is not whether or not to design, the question is when to design**

***Ward Cunningham’s term “technical debt” captures the idea of accumulating design cruft – the eventual consequences of slapdash software architecture and hasty software development...***

### **Ten-minute build**

- **automatically build the whole system and run all of the tests in ten minutes**

28 of 35

## *Topics*

**An Overview of XP**

**XP Management Practices**

**XP Design Practices**

**XP Code and Test Practices**

**→ Summing Up Agile and XP**

29 of 35





## *Principles of Agile Software Development*

*A. Cockburn, "Learning From Agile Software Development"*







- 1) Different projects need different methodology trade-offs.
- 2) A little methodology does a lot of good; after that, weight is costly.
- 3) Larger teams need more communication elements.
- 4) Projects dealing with greater potential damage need more validation elements.
- 5) Formality, process, and documentation are not substitutes for discipline, skill, and understanding.
- 6) Interactive, face-to-face communication is the cheapest and fastest channel for exchanging information.
- 7) Increased communication and feedback reduces the need for intermediate work products.
- 8) Concurrent and serial development exchange development cost for speed and flexibility.
- 9) Efficiency is expendable in non-bottleneck activities.
- 10) Sweet spots speed development.

30 of 35

## *Aligning Agility and Discipline*

-  Early and continuous delivery
-  Changing requirements, even late in development
-  Work with business people (customers and end users)
-  Motivated individuals with support they need . . . *People CMM*

### Face-to-face conversation

-  Working software
-  Sustainable development, constant pace
-  Continuous attention to technical excellence and good design
-  Simplicity
-  Self-organizing teams . . . *CMMI / IPPD*
-  Reflect on how to become more effective, then adjust behavior

31 of 35

## *Barriers to the Success of XP*

**A customer who insists on the big specification...**

**A culture that requires long hours to prove commitment...**

**Projects that are too big (more than about ten programmers)...**

**An environment with a long time to gain feedback (e.g., realistically test the software)...**

**The wrong physical environment (e.g., team members on different floors, not co-located)...**

**We do XP, just not most/any of the practices...**

32 of 35

*Cultural Misfits*  
*(Using the DoD as an Example...)*

**Regulatory requirements for a level playing field raise challenges for evolutionary and incremental development...**

**The need by the contracts officer for a requirements specification...**

**Progress payments defined from a waterfall mentality...**

**Barriers – regulatory and cultural – to a collaborative customer relationship...**

**Protests from competitors...**

33 of 35

*Questions and Answers*



34 of 35

## *Contact Information*

**Dr. Mark C. Paulk**  
**Carnegie Mellon University**  
**407SCR 115**  
**5000 Forbes Avenue**  
**Pittsburgh, PA 15213**

**[mcp@cs.cmu.edu](mailto:mcp@cs.cmu.edu)**  
**[Mark.Paulk@ieee.org](mailto:Mark.Paulk@ieee.org)**  
**<http://www.cs.cmu.edu/~mcp/>**

