

*Panel: Myths, Realities,  
and Best Practices for Agile  
Software Development*

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

*August 12, 2009*  
*San Francisco Bay Alumni Chapter*

*Panelists*

**Bill Cuan, Autonomy**

**Tim Lander, Activision**

**Eric Lee, like.com**

**Emil Ochotta, Yahoo**

**Merline Saintil, Adobe**

## *Agile 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

That is, while there is value in the items on the right, we value the items on the left more.

**Kent Beck**  
**Mike Beedle**  
**Arie van Bennekum**  
**Allstair Cockburn**  
**Ward Cunningham**  
**Martin Fowler**

**James Grenning**  
**Jim Highsmith**  
**Andrew Hunt**  
**Ron Jeffries**  
**Jon Kern**  
**Brian Marick**

**Robert C. Martin**  
**Steve Mellor**  
**Ken Schwaber**  
**Jeff Sutherland**  
**Dave Thomas**

3 of 12

## *Rakitin's Translation*

**Individuals and interactions**  
over processes and tools.

**Talking to people instead of**  
using a process gives us the  
freedom to do whatever we want.

**Working software over**  
comprehensive documentation.

**We want to spend all our time**  
coding. Remember, real  
programmers don't write  
documentation.

**Customer collaboration over**  
contract negotiation.

**Haggling over the details is**  
merely a distraction from the real  
work of coding. We'll work out  
the details once we deliver  
something.

**Responding to change over**  
following a plan.

**Following a plan implies we have**  
to think about the problem and  
how we might actually solve it.  
Why would we want to do that  
when we could be coding?

**S.R. Rakitin, "Manifesto Elicits**  
**Cynicism," IEEE Computer,**  
**December 2001, p. 7.**

4 of 12

## *Three Important (Panel) Questions*

**How has using agile methods helped projects succeed or fail in your company?**

- What's usefully different about agile?

**How do projects measure business success in your company?**

- predictability, short cycle time, customer satisfaction, low cost, innovation, ...

**What factors affected adoption of agile methods?**

- What caused you to consider agile?
- What barriers hindered adoption?
- What tailorings were necessary to fit your culture?

## *Books on Agile Methods*

**B. Boehm and R. Turner, Balancing Agility and Discipline: A Guide for the Perplexed, 2004.**

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

**C. Larman and B. Vodde, Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum, 2008.**

## *Counterpoint Publications*

**M. Stephens and D. Rosenberg, Extreme Programming Refactored: The Case Against XP, 2003.**

**G. Keefer, “Extreme Programming Considered Harmful for Reliable Software Development 2.0,” September 2003.**

- <http://avocallc.com/downloads/ExtremeProgramming.pdf>

**V. Skowronski, “Do Agile Methods Marginalize Problem Solvers?” IEEE Computer, October 2004, pp. 120, 118-119.**

## *The Sweet Spot for Agile Methods*

**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**

## *Scrum*

A process for incrementally building software in complex environments.

**Backlog** – all outstanding work for a product area

**Sprints** – 30-day increments of work that produce a deliverable

**Scrums** – daily status check meetings

*K. Schwaber and M. Beedle, [Agile Software Development with Scrum, 2002.](#)*

*K. Schwaber, [Agile Project Management with Scrum, 2004.](#)  
<http://www.controlchaos.com>*

9 of 12

## *Extreme Programming Practices*

Planning game

Pair programming

Small releases

Collective (code) ownership

Metaphor

Continuous integration

Simple design

40-hour week  
• (sustainable pace)

Testing

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

On-site customer  
• (whole team)

Refactoring

- (design improvement)

Coding standard

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

10 of 12

## *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.](#)*

11 of 12

## *Contact Information*

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

**mcp@cs.cmu.edu**  
**Mark.Paulk@ieee.org**  
**<http://www.cs.cmu.edu/~mcp/>**



12 of 12