Agile Project Management: Scrum

Dr. Mark C. Paulk

SE 4381, Software Project Planning and Management

Jonsson School of Engineering and Computer Science

Management Topics

1. Modern project management

PMBOK

2. Organization strategy and project selection

3. Organization: structure and culture

4. Defining the project

5. Estimating times and costs

6. Developing a project plan

7. Managing risk

8. Scheduling resources and cost

9. Reducing project duration

10. Leadership

11. Teams

12. Outsourcing

13. Monitoring progress

14. Project closure

15. International projects

16. Oversight

17. Agile PM

Critical chain project management

What Is An "Agile Method"?

A software engineering "methodology" that follows the Agile Manifesto?

A method that supports responding rapidly to changing requirements?

- **Mark Paulk**

Does an agile method necessarily imply

- **Evolutionary / iterative / incremental development?**
- **Empowerment / participation of the development team?**
- **Active collaboration with the customer?**
- **…**

An Agile Timeline

1990s	**Lightweight methods…**
1995	**Scrum**
1996	**Extreme Programming (XP)**
1999	**Beck, Extreme Programming Explained**
2000	**Paulk, "XP from a CMM Perspective" in eXtreme Programming Pros and Cons: What Questions Remain? J. Siddiqi (ed)**
2001	**Agile Manifesto (Snowbird meeting)** **Ambler and Jeffries, Agile Modeling** **Paulk, "Extreme Programming from a CMM Perspective," IEEE Software**

2002	Schwaber and Beedle, Agile Software Development with Scrum
	Paulk, "Agile Methodologies and Process Discipline," Crosstalk
2003	Stephens and Rosenberg, Extreme Programming Refactored
2004	Beck and Andres, Extreme Programming Explained, 2nd Edition
	Boehm and Turner, Balancing Agility and Discipline
	Cockburn, Crystal Clear
	Larman, Agile and Iterative Development: A Manager's Guide
2007	Kniberg, Scrum and XP from the Trenches

Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

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
Alistair 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

Principles Behind the Agile Manifesto

Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

Business people and developers must work together daily throughout the project.

Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.

The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

Working software is the primary measure of progress.

Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

Continuous attention to technical excellence and good design enhances agility.

Simplicity--the art of maximizing the amount of work not done--is essential.

The best architectures, requirements, and designs emerge from self-organizing teams.

At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Cockburn's Principles of Agile 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.**

A. Cockburn, "Learning From Agile Software Development," Crosstalk: The Journal of Defense Software Engineering, October & November 2002.

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.**

Six Sweet Spots

Dedicated developers

Experienced developers

Small co-located team

Automated regression tests

Easy access to users

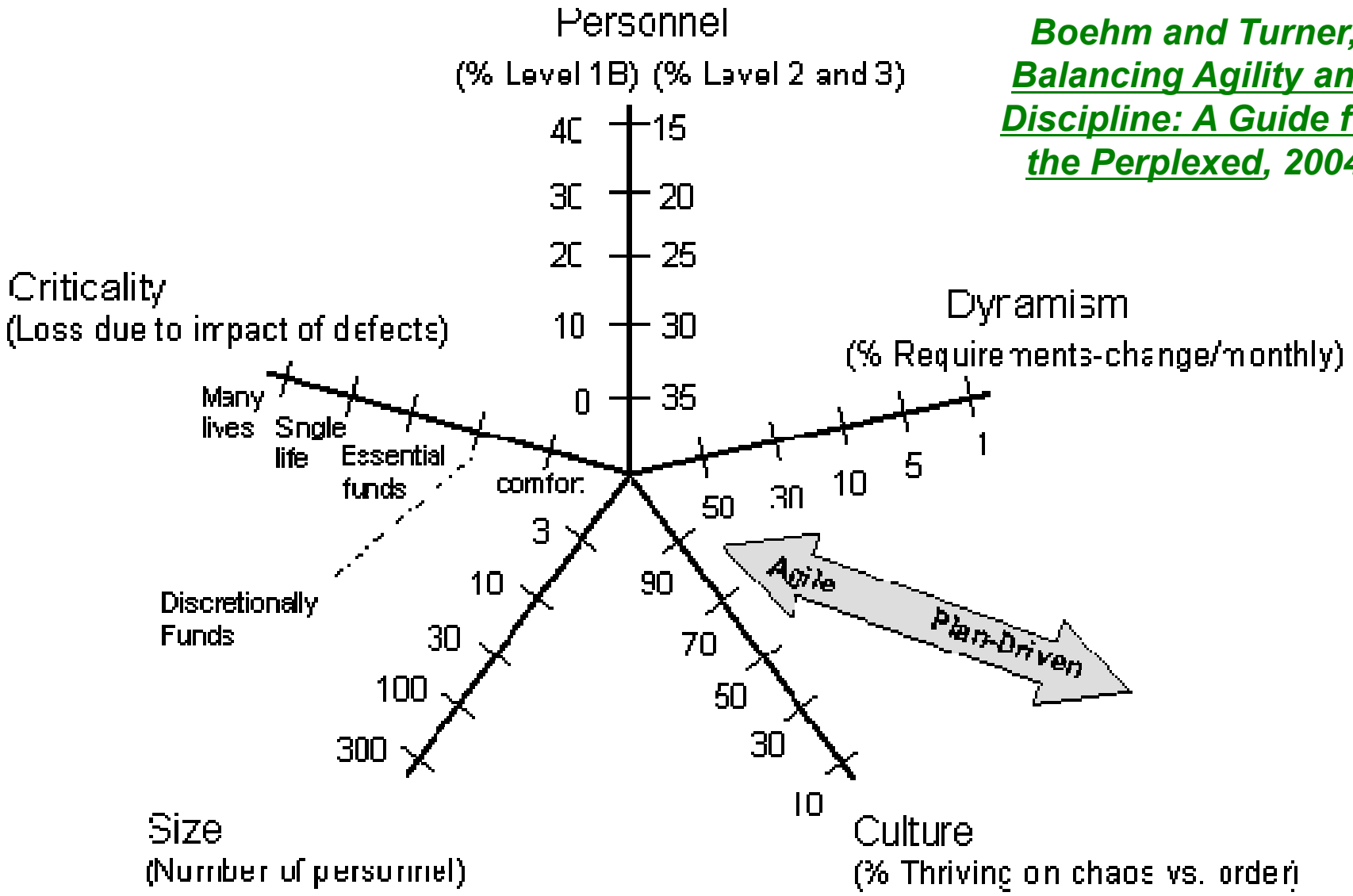
Short increments and frequent delivery to real users

Is Agile an Excuse for Hacking?

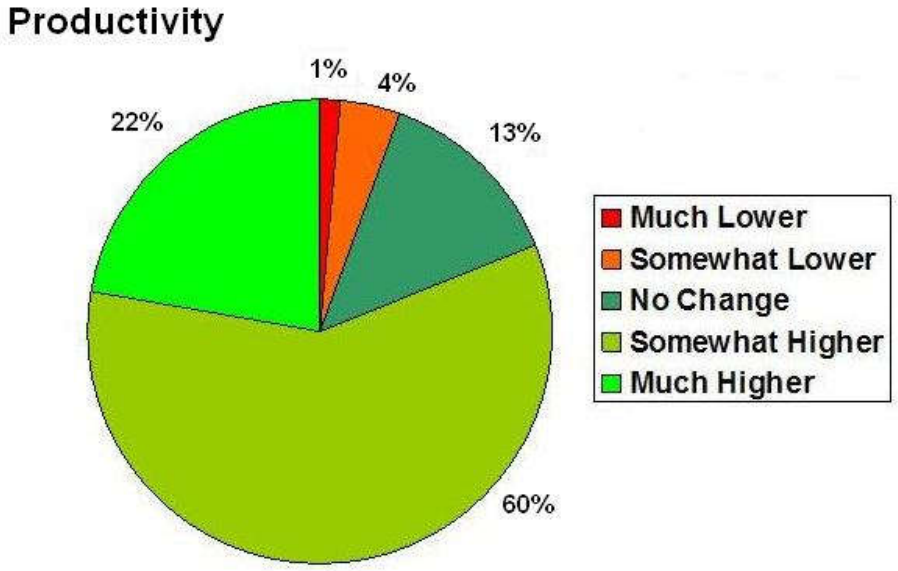
Individuals and interactions over processes and tools.	**Talking to people instead of using a process gives us the freedom to do whatever we want.**
Working software over comprehensive documentation.	**We want to spend all our time coding. Remember, real programmers don't write documentation.**
Customer 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?**

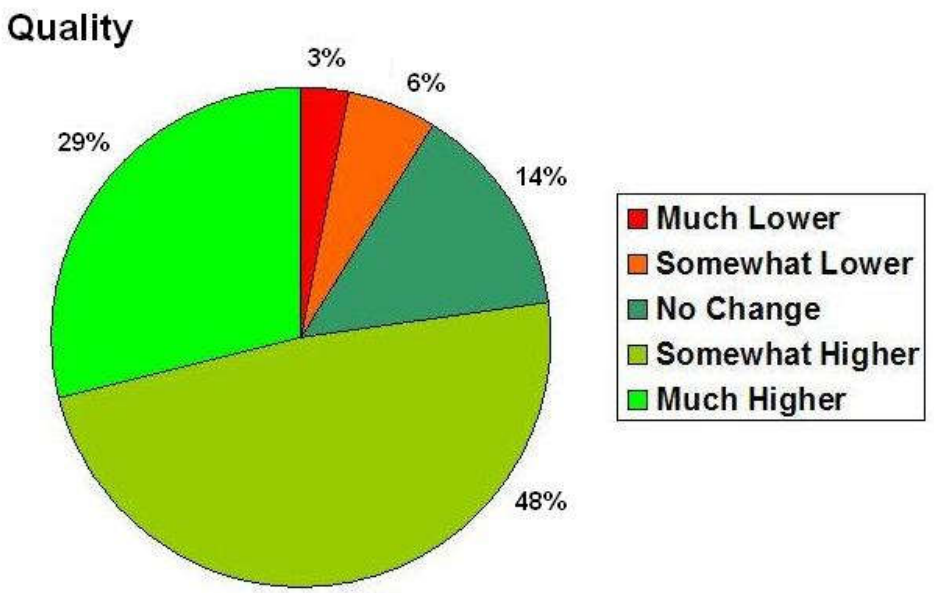
S.R. Rakitin, "Manifesto Elicits Cynicism," IEEE Computer, December 2001, p. 7.

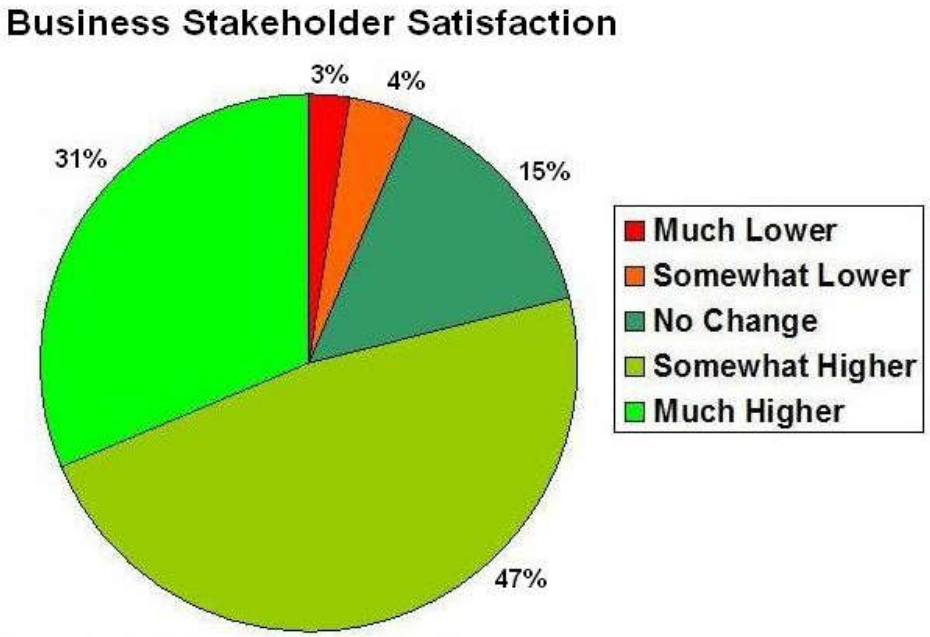
Does Agile Fit Your Needs?

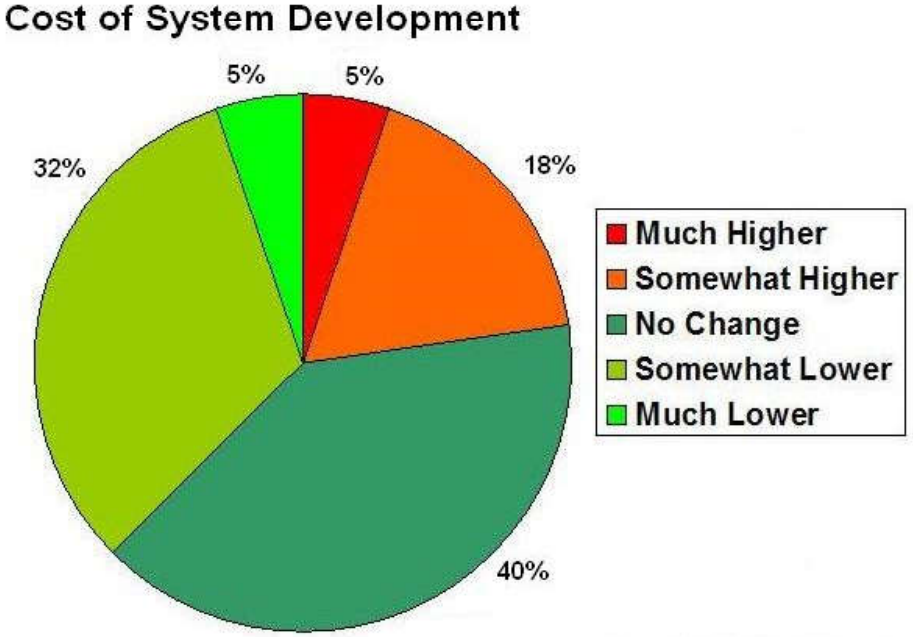


Boehm and Turner, Balancing Agility and Discipline: A Guide for the Perplexed, 2004.









Source: Dr. Dobb's Journal 2008 Agile Adoption Survey

S.W. Ambler, "Agile Adoption Rate Survey Results: February 2008," <URL: http://www.ambysoft.com/surveys/agileFebruary2008.html>

Important Agile Methods

Scrum

Extreme Programming (XP)

Crystal Light Methods, specifically Crystal Clear

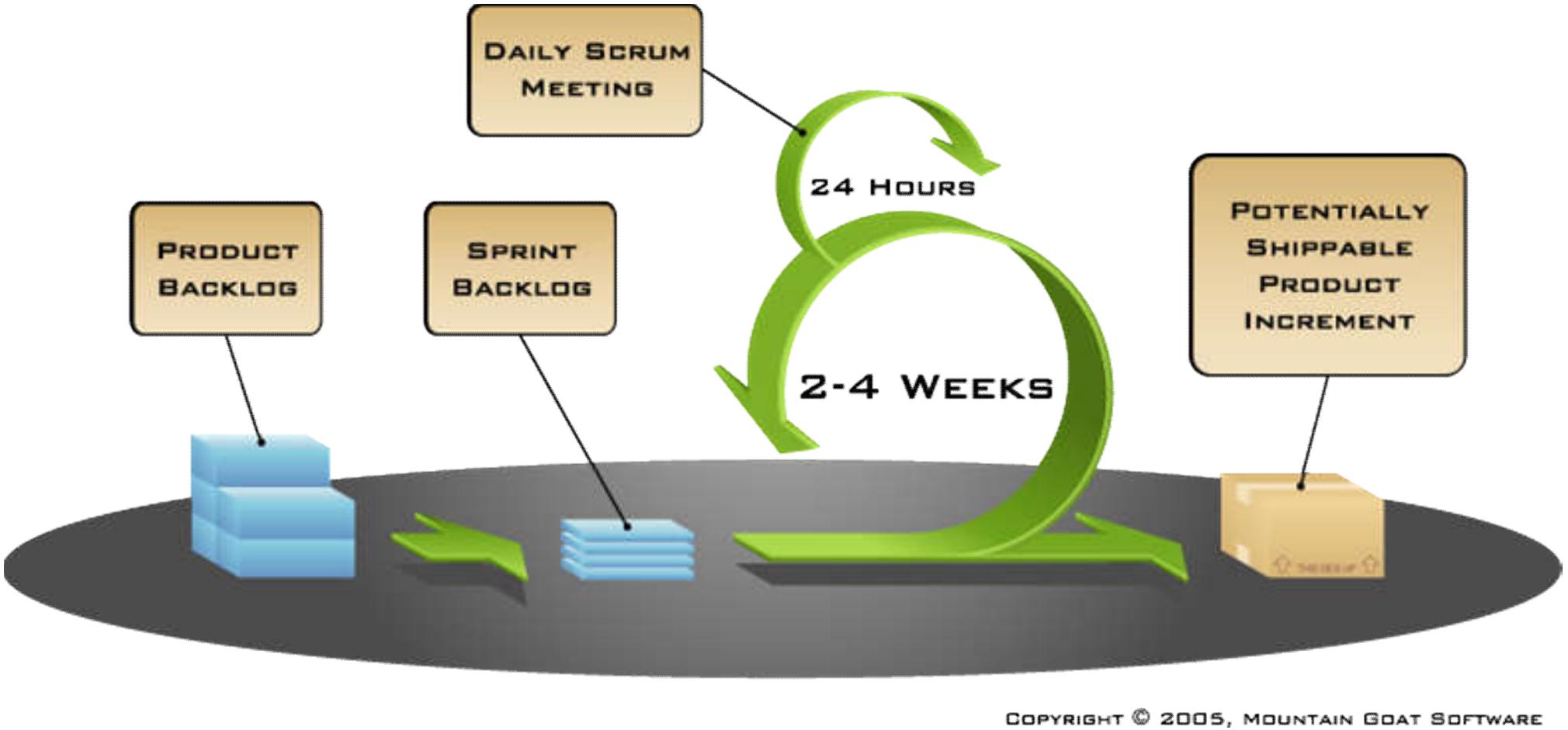
Feature Driven Development (FDD)

Lean Development

Kanban (Scrumban)

and so forth…

Scrum – Inspect & Adapt (Cohn 2010)



Product Owner (James 2010)

- **Single person responsible for maximizing the return on investment (ROI) of the development effort**
- **Responsible for product vision**
- **Constantly re-prioritizes the Product Backlog, adjusting any long-term expectations such as release plans**
- **Final arbiter of requirements questions**
- **Accepts or rejects each product increment**
- **Decides whether to ship**
- **Decides whether to continue development**
- **Considers stakeholder interests**
- **May contribute as a team member**
- **Has a leadership role**

ScrumMaster (James 2010)

- **Facilitates the Scrum process**
- **Helps resolve impediments**
- **Creates an environment conducive to team self-organization**
- **Captures empirical data to adjust forecasts**
- **Shields the team from external interference and distractions to keep it in group flow (a.k.a. the zone)**
- **Enforces timeboxes**
- **Keeps Scrum artifacts visible**
- **Promotes improved engineering practices**
- **Has no management authority over the team**
  - **anyone with authority over the team is by definition not its ScrumMaster**
- **Has a leadership role**

Scrum Development Team (James 2010)

- **Cross-functional (e.g., includes members with testing skills, and often others not traditionally called developers: business analysts, domain experts, etc.)**
- **Self-organizing / self-managing, without externally assigned roles**
- **Negotiates commitments with the Product Owner, one Sprint at a time**
- **Has autonomy regarding how to reach commitments**
- **Intensely collaborative**
- **Most successful when located in one team room, particularly for the first few Sprints**
- **Most successful with long-term, full-time membership.**
  - **Scrum moves work to a flexible learning team and avoids moving people or splitting them between teams.**
- **7 ± 2 members**
- **Has a leadership role**

Product Backlog

Lists the requirements for the product being developed

The master list of all functionality desired in the product

Each item in the Product Backlog has a description, a priority, and an estimate of the effort needed to complete it.

Division of Concerns

During a Sprint Planning Meeting, scope, importance, and estimates are tuned continuously through face-to-face dialog between the Development Team and the Product Owner.

Scope and importance are set by the Product Owner.

Estimates are set by the Development Team.

What's a User Story?

A promise to have a discussion; not every detail needs to be included.

Describes functionality that will be valuable to either a user or purchaser of a system.

- ***Card*** **– written description of the story used for planning and as a reminder**
- ***Conversation*** **– about the story that serve to flesh out the details of the story**
- ***Confirmation*** **– details that can be used to determine when a story is complete**

Ron Jeffries, XP Magazine, August 30, 2001.

Story Points

People are better at estimating "this is like that" than they are at absolute values.

Story point estimates do not change because of experience.

When estimates relative to estimates for other features, it does not matter whether they are correct, high, or low, so long as they are consistent… velocity is the equalizer.

Planning Poker

Everyone on the Team is involved in estimating every story – builds common understanding, collective ownership.

Deck of cards: 0, ½, 1, 2, 3, 5, 8, 13, 20, 40, 100, ?

Simultaneous estimation by facing card with estimate in story points

- **a variation of the wideband Delphi technique (Boehm 1981)**

Anchor with a typical story worth 5 story points

Sprints

A development iteration of a month or less in duration.

Sprint duration is fixed throughout the development effort.

Only the Product Owner has the authority to cancel the Sprint.

Sutherland and Vodde suggest that Sprints may be 2-6 weeks long.

- ***J. Sutherland and B. Vodde, "The Nokia Test, Extended with Scoring by Jeff Sutherland," 2008.***

Sprint Planning Meeting

When the iteration (Sprint) is planned

Time boxed to eight hours (for a one month Sprint)

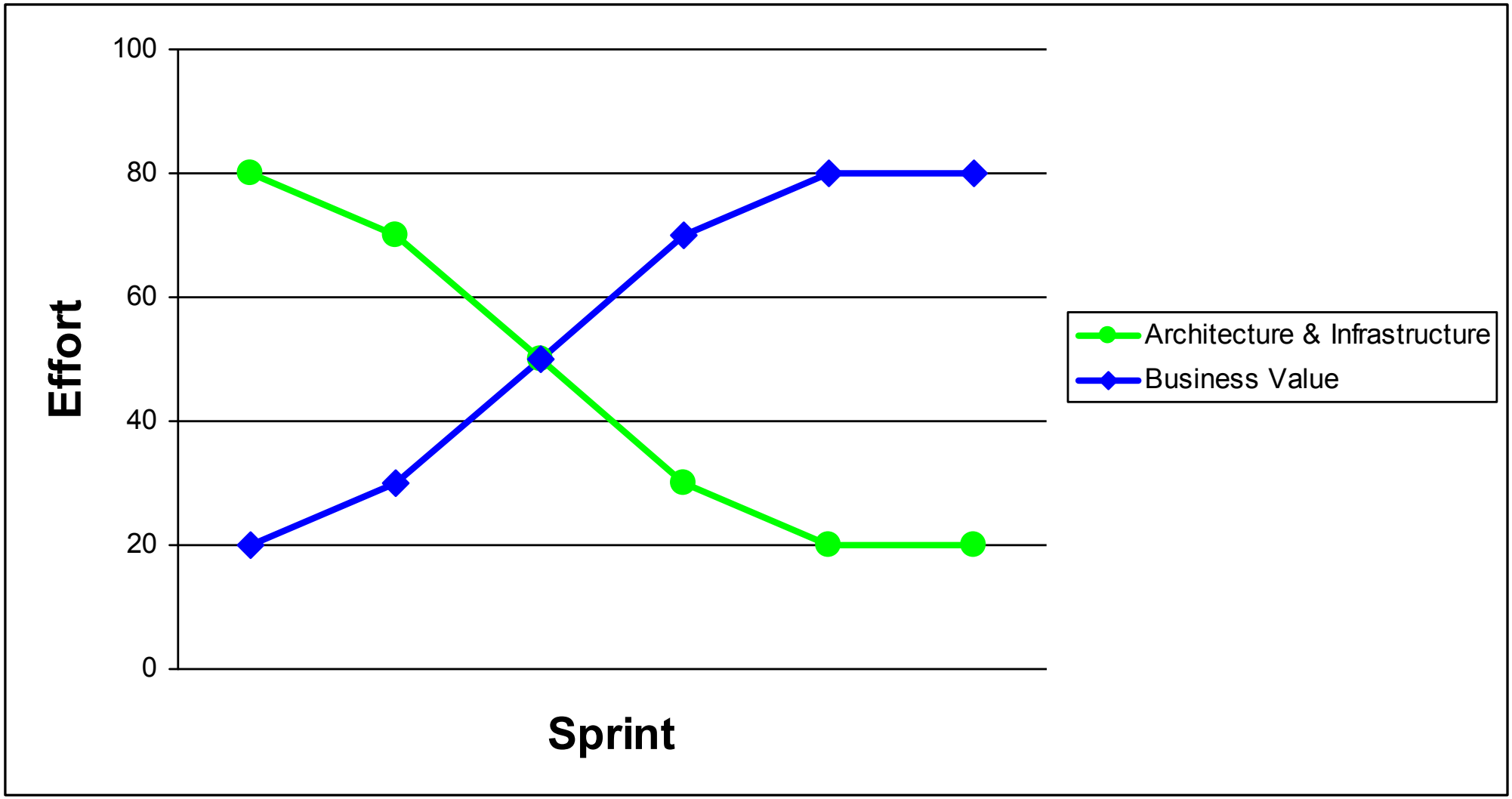
Two parts (4+4 hours):

- **determining what will be done in the Sprint**
- **how the Team is going to build the product increment during the Sprint**

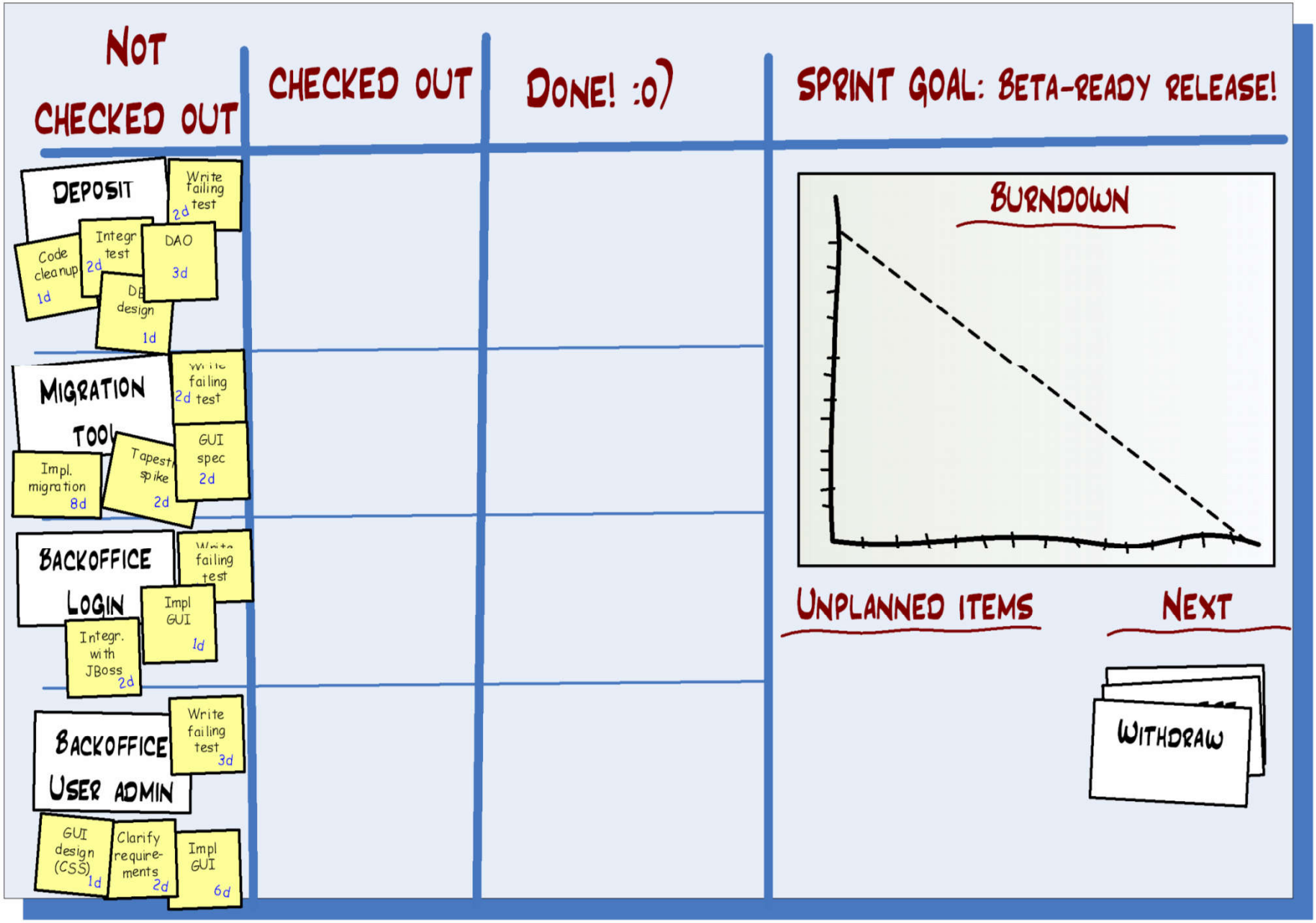
Output is the Sprint Backlog

- **requirements to be addressed in this Sprint**

Building the Foundation



A Taskboard (Kniberg 2007)



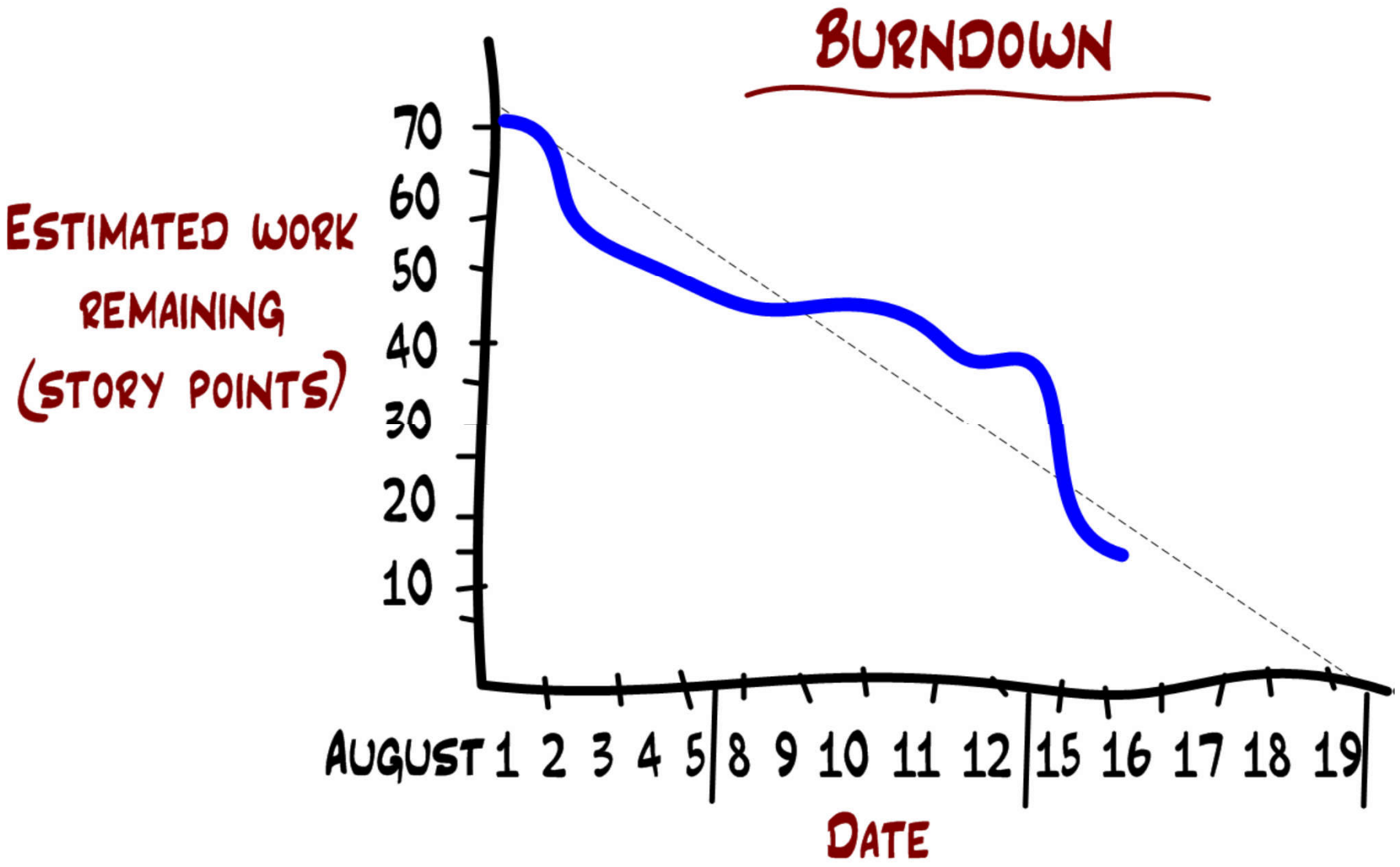
Daily Scrum Meeting

A time-boxed, 15-minute meeting used to inspect progress toward the Sprint goal and to make adaptations that optimize the value of the next workday.

Three questions:

- **What did you do since the last Scrum?**
- **What got in your way?**
- **What are you going to do before the next Scrum?**

A Burndown Chart (Kniberg 2007)



Sprint Review Meeting

A four-hour time-boxed meeting (for one-month Sprints) that is held at the end of a Sprint where the Team presents the functionality done in the iteration to the Product Owner and other stakeholders.

The Team demonstrates and discusses the work done in the Sprint.

Sprint Retrospective

A three hour, time-boxed meeting (for one-month Sprints) held after the Sprint Review and prior to the next Sprint Planning meeting where the Team discusses what went well in the last Sprint and what can be improved for the next Sprint.

I'd say the retrospective is the second most important event in Scrum (the first being the Sprint Planning Meeting).

- ***H. Kniberg, Scrum and XP from the Trenches, 2007.***

No Silver Bullet

Scrum will not solve your problems.

Scrum will make your problems visible.

You will have to solve your problems.

XP Essentials

XP assumes the high cost of change has been addressed by technologies such as objects/patterns, relational databases, information hiding, etc.

XP improves a software project in four essential ways: communication, simplicity, feedback, and courage

Four basic activities of XP: coding, testing, listening, and designing

K. Beck, Extreme Programming Explained: Embrace Change, 1999.

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/whatisxp.htm)

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.

Barriers to the Success of Agile

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 "agile", just not most / any of the practices…

Questions and Answers