

For those who were Agile before Agile was cool

Dr. James O. Coplien

Senior Agile Coach

Gertrud & Cope, Denmark

What is “Agile Development?”



- 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.

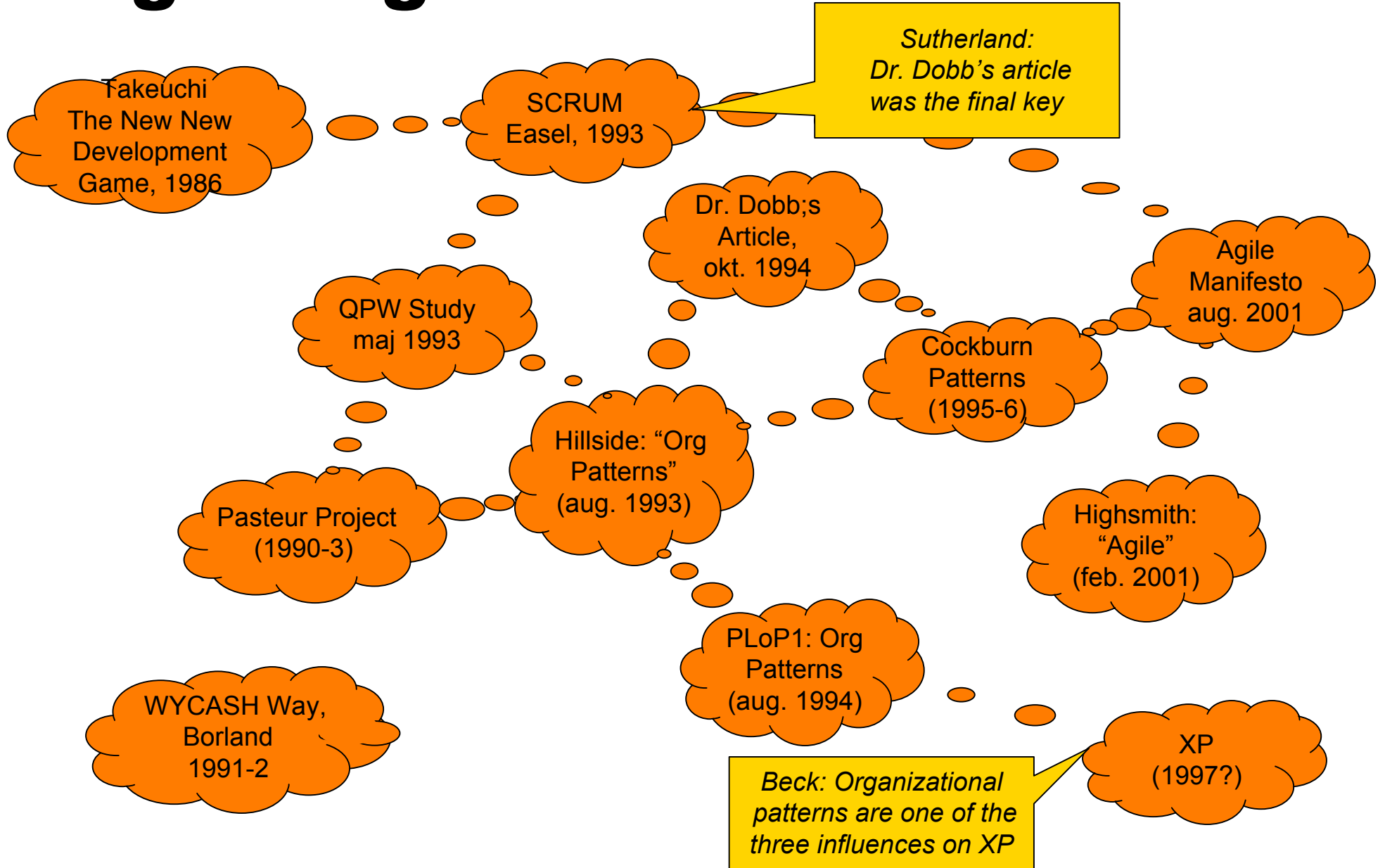
- Kent Beck
- Mike Beedle
- Arie van Bennekun
- Alistair Cockburn
- Ward Cunningham
- Martin Fowler

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

- Robert Cecil Martin
- Steve Mellor
- Ken Schwaber
- Jeff Sutherland
- Dave Thomas

© 2001, the above authors
this declaration may be freely copied in any form,
but only in its entirety through this notice.

Beginnings



Short History of eXtreme Programming



From: Kent Beck
To: Jeff Sutherland <jsutherland>
Reply: 70761.1216@compuserve.com
Date: Mon, 15 May 1995 18:01:15 -0400 (EDT)
Subj: HBR paper

Is there a good place to get reprints of the SCRUM paper from HBR? I've written patterns for something very similar and I want to make sure I steal as many ideas as possible.
Kent

Agile: The Foundations



- 1960s MIT Hackers
- MVC, 1978
- Boehm's Spiral Development, 1986
- The Software Pattern Discipline, 1993
- Cathedral & Bazaar, 1997

Hacking (a.k.a “Real Programmers”)



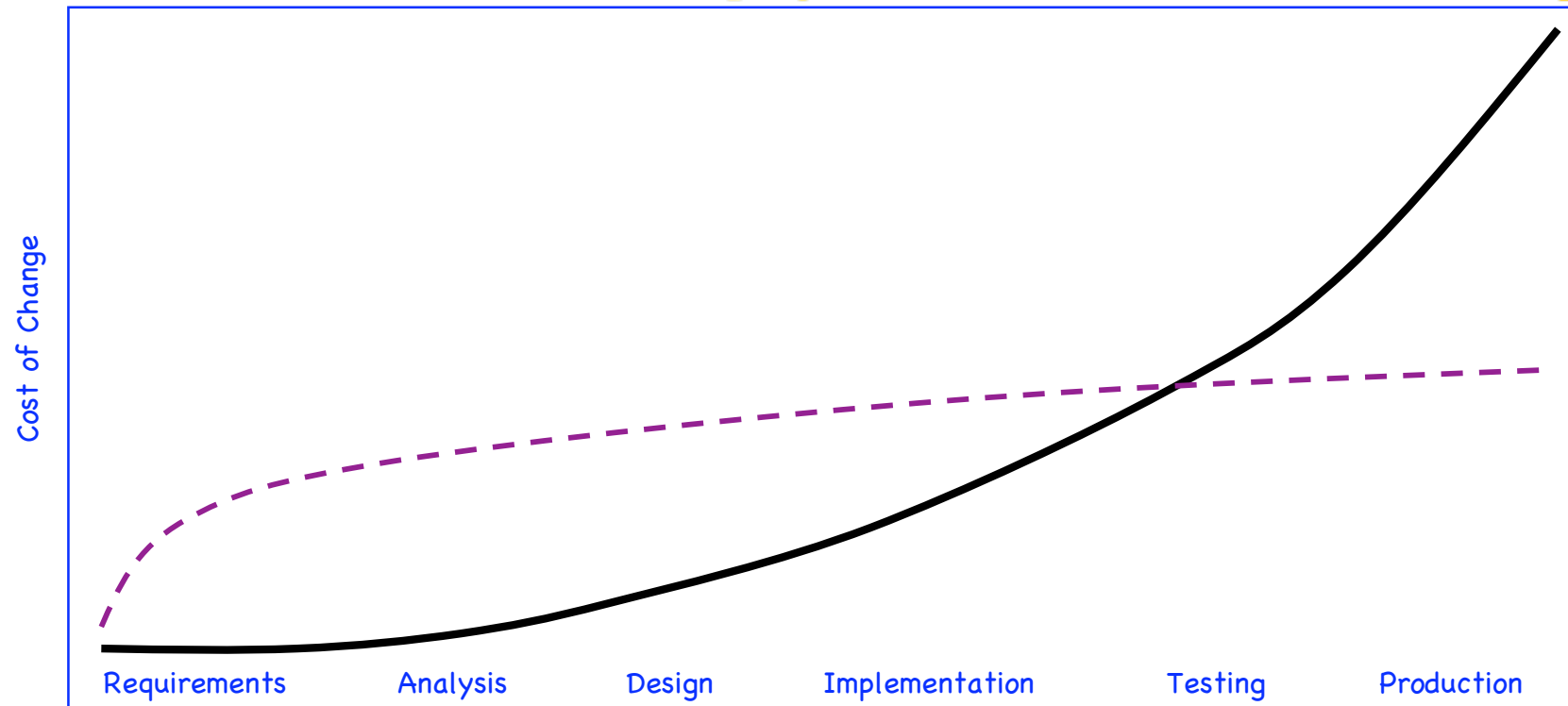
- A kind of “goofing off” or “play” form
- Exploring boundaries, opening closed doors
- 1950s roots
- 1960s — model railway control geeks using the TX-0
- Spacewar, DEC PDP-1
 - Computer as “representing action in which humans can participate” — Brenda Laurel
- Disdain for methods
- Courage of XP: hackers were confident
- Individuals and Interactions — this was for fun, not business—the first *social* example of hacking

What really has been happening the past 20 years?

- Start Small — don't presume to know all requirements up front, and don't master plan, but start with a small subset of requirements, a small system and, most importantly, a small number of staff
- Iterative development — If one has started small, one can grow incrementally and iteratively. Build a series of fully functional systems in rapid succession and use them to gain understanding of requirements, to engage the customer in the process, and to build piecemeal. Manage change and manage for change rather than managing to a supposedly known set of requirements
- Use Cases — Capture Joint Stories, Joint Scenarios and formalized collections of scenarios called Use Cases, and use those, rather than formal requirements documents, to drive the process. Turn Use Cases and Joint Scenarios into test scripts early in the project and use these to drive development. Get Continuous Requirements updates.
- Develop Good Enough — not too much for the future, not too pretty, just good enough to not lose the market.
- Problem Statement — find, articulate and know what problem you are solving.



It doesn't add up



- ...but the doubt about the Standish report (Robert Glass, Moløkken-Østvold, Jørgensen, ...)
- Many practices (not the Manifesto itself) derive from myths!
- This is the sole technical basis of XP, and it is groundless

MVC, 1978

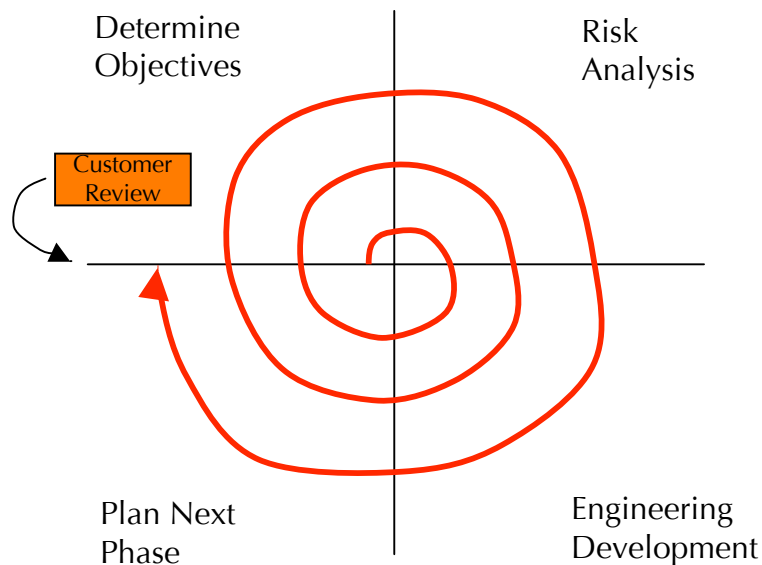


- “MVC was conceived in 1978 ... the top level goal was to support the user’s mental model of the relevant information space and to enable the user to inspect and edit this information”
- Every part of MVC is “motivated by the needs of people and the desire to create habitable information systems.”
- Brenda Laurel, 1991: Direct Manipulation Metaphor
- Individuals and Interactions over GUI-builders
 - You cannot build a humane interface using an interface builder — Jef Raskin
- Working (in the sense of the interface) software instead of documentation
- Customer collaboration (by the program) over contract negotiation

Spiral, 1986

- Boehm, 1986, refined in 1988
- Take a handful of requirements at a time
- Feedback, feedback, feedback

- Customer Collaboration over contract negotiation — do a little at a time
- Responding to Change over following a Plan
- Note that the 4 values of the Manifesto say nothing about iterative development!!!



Patterns, 1993



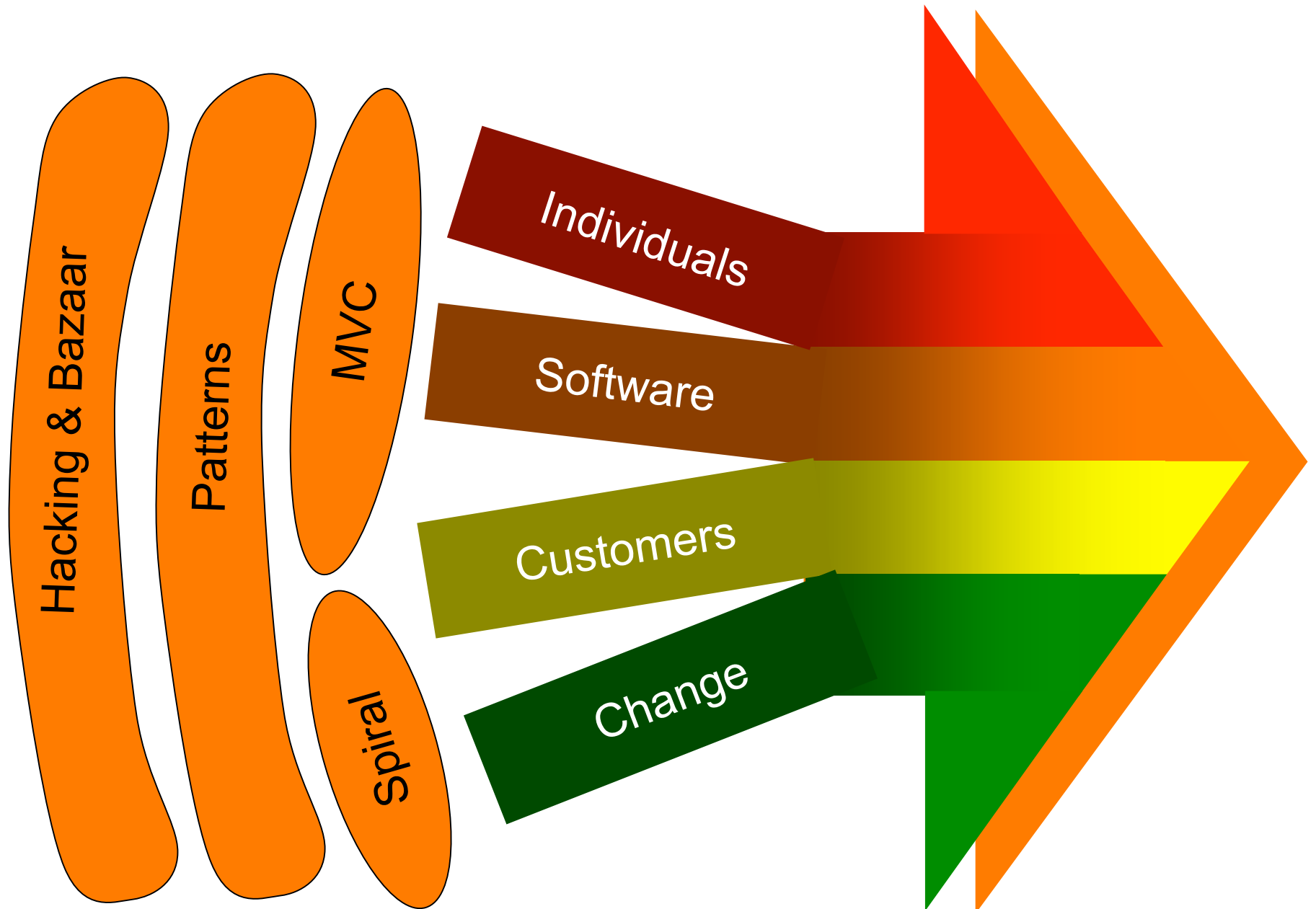
- Architecture
- Focus on the user
 - Beauty (in GUIs)
 - Beyond MVC: A belief that beauty in GUIs had to come from beauty in Code
- Beauty, not business
- Systems Thinking
- Org Patterns: Adaptive process definition
- Individuals and Interactions over Processes and Tools
- Working software
 - Alexander: No blueprints
- Customer Collaboration
 - Alexander: Ideally, the dwellers build the house
- Responding to Change
 - Everything just-in-time

Bazaar, 1997



- Release Early and Often
- Delegate Everything you Can
- Total openness
- Ten principles:
 1. Your personal itch
 2. Write and rewrite
 3. Plan to throw one away
 4. Attitude
 5. Ownership succession
 6. Users as co-developers
 7. Release early, often, get feedback
 8. Beta Testing and co-development
 9. Smart Data Structures
 10. Valuing Beta Testers
- Responding to Change
- Individuals and Interactions over processes and tools
- Customer Collaboration

Towards the Agile Manifesto



MVC: where have all the flowers gone?



- Support user *mental* model
- Habitable information systems
- Focus on the user
- Working *software* — meaning code
- YAGNI and no architecture
- Focus on GUI-builders

Spiral: It's all there...

and more

- 
- A little at a time
 - Respond to change
 - Incremental
 - Embrace Change
 - Iterative!

Patterns: where have all the flowers gone?

- User focus
- Beauty
- Sketches, not Blueprints
- *Systems* Thinking
- Architecture
- Adaptive process
- Customer (business) focus
- Functionality
- Not sketches either
- “...over comprehensive documentation”
- *Unit* test
- *Pair* programming
- *Individuals* and interactions
- YAGNI
- Responding to change over following a plan
- Prescribed practices

Bazaar: where have all the flowers gone?



- Write and rewrite
- Early/often/feedback
- Pride in Ownership, and a duty to find a successor
- Co-developing with users
- Beta Testing
- Refactoring
- Yes!
- Collective ownership
- On-site customer
- TDD for *testing* (Beck)
- TDD for *design* (Jimmy Nilsson)

Back to Modernism



■ Profit and opportunism

- Patterns proved unprofitable in 1994: repackaged into things that would appear three years later as a marketable entity
 - Mail from a Hillsider, April 1994: “Fact is, my patterns effort have been cutting into revenue and it can’t continue”
- The breaking of community: pattern folks went off into more commercially oriented outlets
 - Mail from the same Hillsider in May 1995: scrambling to package Scrum ideas into a personally branded commercial framework
- The sole technical basis of XP is groundless

■ Nurdism

- TDD *tools* — no notion of testing against customer needs
- Use tests as documentation instead of rising above the code

Future directions

- Individuals and interactions over processes and tools
 - More focus on the interface, less on the program
 - Trygve Reenskaug: We need more research on architecture for the sake of the *interface*!
- Working software over comprehensive documentation
 - To the user, software is documentation and the GUI
 - The interface is the program — Jef Raskin
 - System testing
- Customer collaboration over contract negotiation
 - Gabriel's vision of swarm software
 - Back to the vision of Hackers and of the Bazaar
- Responding to change over following a plan
 - Free people to change the process — no more religions
 - More domain-driven design: Foundations for flexibility and customer engagement
- Back to the agendas of people — and fun!

Why will it fail?



- Pekka Abrahamsson: Agile is conceptually flawed
- Juha-Pekka Tolvanen: Processes and tools over people and interactions
- Frederik Ferm: So, how do you know a team is Agile? By the post-its on their... screen...
- Matti Antila, Microsoft: It's about people

The 00 Days Manifesto



- We are uncovering better ways of selling product by appealing to the unfeeling nerd in all of us. Through this work we have come to value:
 - Processes and tools over individuals and interactions
 - Software that works for us even if our users can't figure it out
 - Contract negotiation (see us if you want to learn more) over customer collaboration
 - Defining the direction of change so that we'll be in front
- That is, while there is value in the items on the right, whatever we do, we'll call Agile.

History again

