Agile software development

Bertrand Meyer

Part A: context

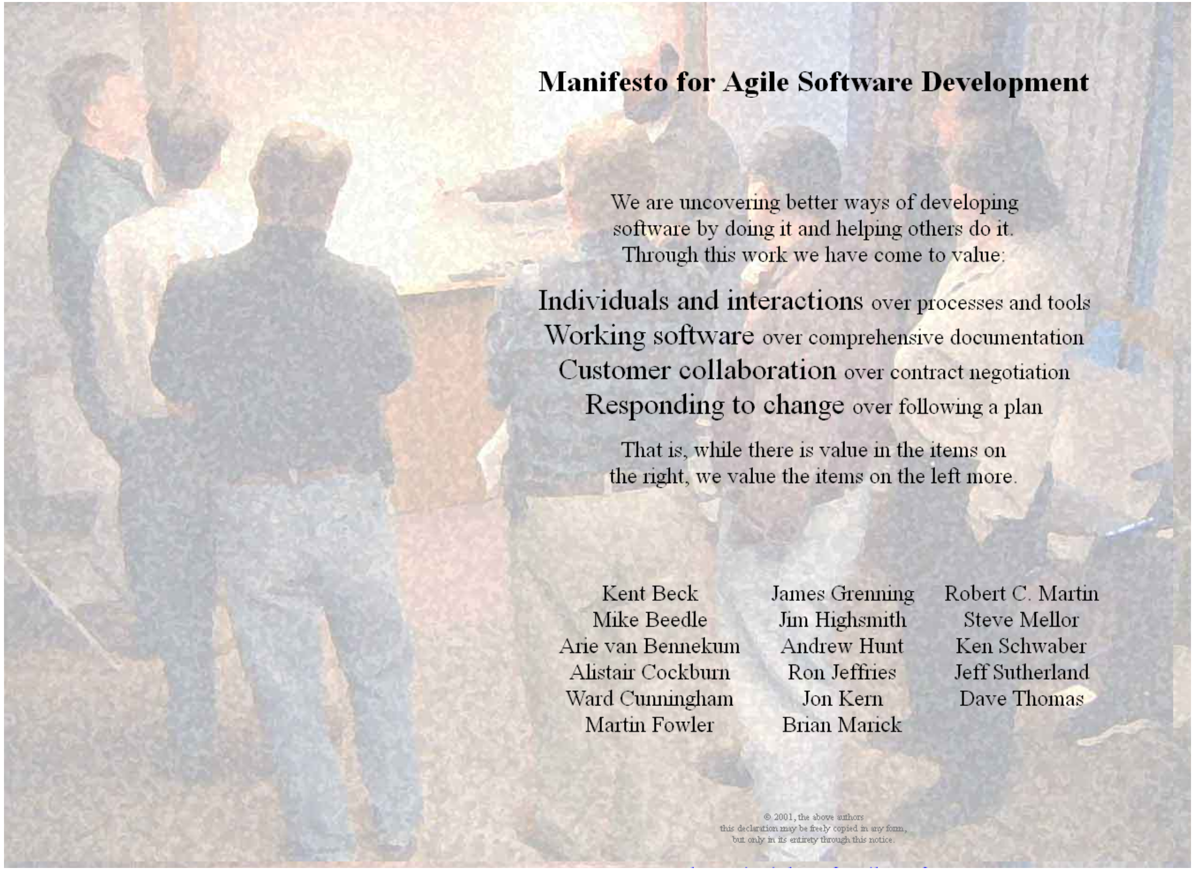
1: The Agile Manifesto

2: Agile methods

3: Official agile principles

4: Agile values

Agile manifesto



Manifesto for Agile Software 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 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

agilemanifesto.org

© the authors

Twelve principles (from the Manifesto)

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity— the art of maximizing the amount of work not done —is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile Software Development

Bertrand Meyer

Part A: Context

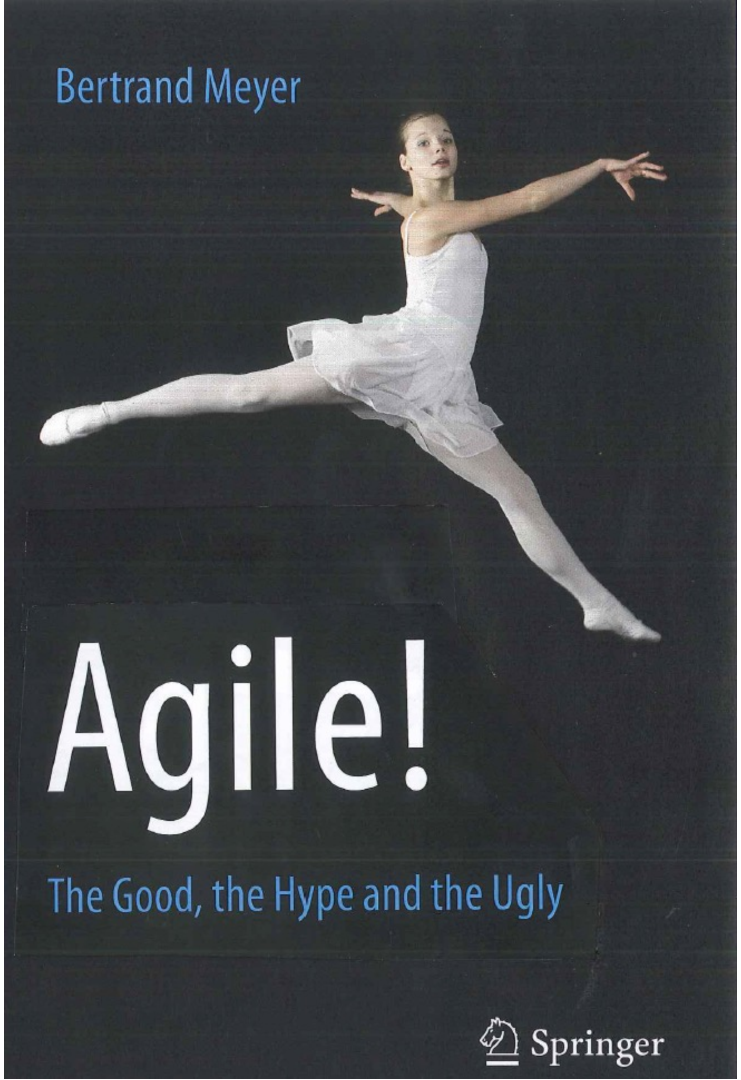
1: The Agile Manifesto

What we have seen:

A set of principles,

defining Agile as seen by its creators

Supporting textbook



Agile!
The Good, the Hype and the Ugly

Bertrand Meyer
Springer, 2014

Agile software development

Bertrand Meyer

Part A: context

1: The Agile manifesto

2: Agile methods

3: Official agile principles

4: Agile values

Agile methods

XP	(Extreme programming) Kent Beck
Lean	Mary Poppendieck
Crystal	Alistair Cockburn
Scrum	Schwaber & Sutherland

XP: The revolt of the cubicles

Negotiated scope contract

XP

Source: Beck 05

“Write contracts for software development that fix time, costs, and quality but call for an ongoing negotiation of the precise scope of the system. Reduce risk by signing a sequence of short contracts instead of one long one.”

“You can move in the direction of negotiated scope. Big, long contracts can be split in half or thirds, with the optional part to be exercised only if both parties agree. Contracts with high costs for change requests can be written with less scope fixed up front and lower costs for changes”

Agile Software Development

Bertrand Meyer

Part A: Context

2: Agile methods

What we have seen:

A number of agile methods have emerged

They share the basics

but differ in their goals and emphases

ETH zürich

Agile software development

Bertrand Meyer

Part A: context

1: The Agile manifesto

2: Agile methods

3: Official agile principles

4: Agile values

Chair of Software Engineering

Twelve principles

Source: Agile manifesto

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must work together daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity — the art of maximizing the amount of work not done — is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Redundancy

Practice

What about testing?

Redundancy

Assertion

Assertion

Assertion

Wrong

Practice

Finishing a design

It seems that the sole purpose of the work of engineers, designers, and calculators is to polish and smooth out, lighten this seam, balance that wing until it is no longer noticed, until it is no longer a wing attached to a fuselage, but a form fully unfolded, finally freed from the ore, a sort of mysteriously joined whole, and of the same quality as that of a poem. It seems that perfection is reached, not when there is nothing more to add, but when there is no longer anything to remove.

(Antoine de Saint-Exupéry, *Terre des Hommes*, 1937)

Steve Jobs, 1998

That's been one of my mantras — focus and simplicity. Simple can be harder than complex: You have to work hard to get your thinking clean to make it simple. But it's worth it in the end because once you get there, you can move mountains.

Agile Software Development

Bertrand Meyer

Part A: Context
3: Official agile principles

What we have seen:

The official principles give us a good starting point
for understanding Agile,
but they fall short of explaining exactly what the method is

Agile software development

Bertrand Meyer

Part A: context

1: The Agile manifesto

2: Agile methods

3: Official agile principles

4: Agile values

A more precise description

Agile values

Agile principles

- Managerial
- Technical

Agile roles

Agile practices

- Managerial
- Technical

Agile artifacts

Agile values

- **A** New, reduced role for manager
- **B** No “Big Upfront” steps
- **C** Iterative development
- **D** Limited, negotiated scope
- **E** Focus on quality, achieved through testing

Agile Software Development

Bertrand Meyer

Part A: Context

4: Agile values

What we have seen:

Beyond specific principles, practices and artifacts,
Agile is defined by general values,
representing a vision of how development should proceed