Agile Development

What is this all about?

The Agile Manifesto

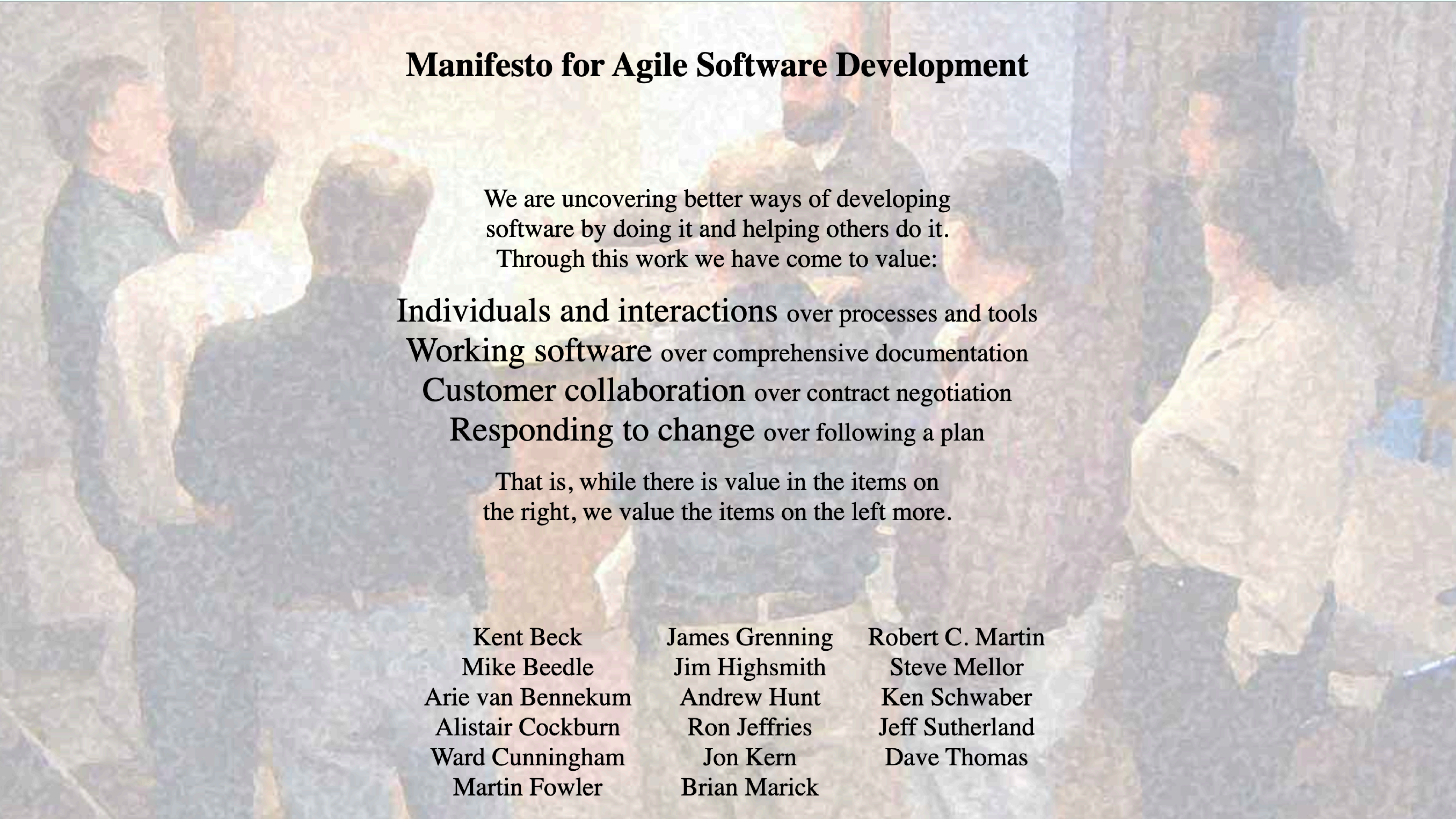
Emerged from a small gathering of programmers at a ski resort in Utah in Feb 2001.

In their experience, software development was inefficient and frustrating. They met to envision and describe a better way.

"Waterfall" development

One big project that proceeds from start to finish in a structured way without changes.

- clearly defined requirements
- plan, design, and document a solution
- write code
- test and make sure it meets requirements
- deliver/deploy
- maintain



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

Individuals and interactions

Developers are trusted to do the right thing in a supportive environment of close collaboration and constant communication.

Working software

Showing software that does something is better than just talking about it during meetings.

The only real measure that counts is whether software works.

Customer collaboration

Involving customers and business people throughout the process is essential to the right outcome.

Responding to change

If something changes (the situation, the requirements, the timeline, or anything), the development teams is flexible and can adjust what they are doing.

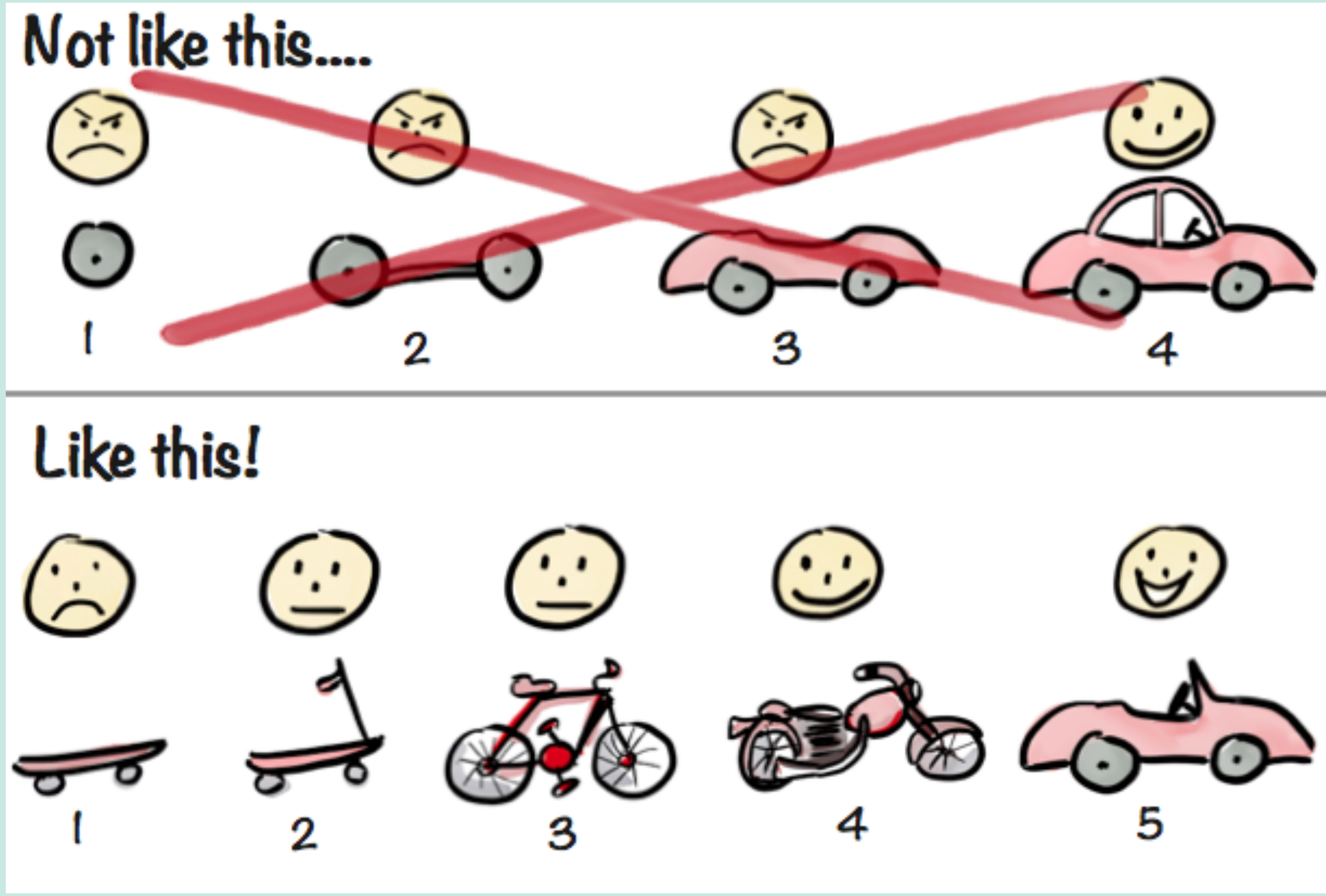
12 Principles

1. Improve customer satisfaction by delivering software early and continuously.
2. Be receptive to changing requirements throughout the development process.
3. Frequently deliver *working* software.
4. Developers should collaborate daily with stakeholders.
5. Support motivated individuals.
6. Face-to-face communication is always best.
7. Measure progress by the delivery of working software.
8. Stakeholders, developers and users should be able to maintain a constant pace.
9. Focused and continuous attention to technical excellence and good design.
10. Simplicity is essential.
11. Self-organizing teams produce the designs, requirements, and architectures.
12. Regularly self assess the team in order to become more effective and to adjust to new circumstances.

The Players

- **Product owner** represents the business (`stakeholder`) point of view to the team
- **Stakeholders** business people, often non-technical, who have a vested interested in the outcome of the project
- **Team** a group of technical professionals working together to produce a product (developers, designers, UI/UX people, QA testers)

MVP



Iterative Development

A model of software development that involves repeated cycles of work to incrementally deliver a working product.

A single iteration goes from product requirements through design, coding, and testing.

The end result is the delivery of complete components of a software product.

Scrum

A set of practices and methods that apply Agile principles to organize and manage a project.

Sprint

A time-limited, single cycle (or iteration) of the development process of a fixed, short length.

A sprint often starts with a sprint planning meeting, ends with a review, and may be followed by a sprint retrospective meeting.

During a sprint, planned tasks are completed.

Backlog

A collection of `tasks` and `stories` to be completed in a given sprint or over several sprints.

Developers choose, or are assigned, tasks from this list to work on.

User Story

A basic unit of the development process that describes a single aspect of the product.

The way it is expressed allows for clear communication and planning between the technical and non-technical people involved.

As a < *type of user* >,
I want to < *do something* >
so that < *some reason* >.

As a *student*,
I want to *see the results of all my past evaluations*
so that *I can see my progress*.

As an *instructor*,
I want to *schedule an evaluation*
so that *it can be completed by students*.

Task

A small, discrete description of work to needs to be done in order to complete a *story*.

One developer, or one pair-programming duo of developers, would work on a single task.

Task board

A visual representation (like Trello) or chart of tasks used to organize a team's work by using categories to represent a stage in the process.

A minimal structure would include 3 columns:

- to do
- in progress
- done

Standup

A brief, daily team meeting in which `story` progress is discussed. Each person has a turn to speak and addresses the following:

1. What they have done since yesterday
2. What they are planning to do today
3. What obstacles are blocking their progress

what do I do with all this information?

This is you.