



Introduction to Agile

What is Agile Project Management?

Department of Computer Engineering
Sharif University of Technology
Maryam Ramezani
maryam.ramezani@sharif.edu





01

Introduction

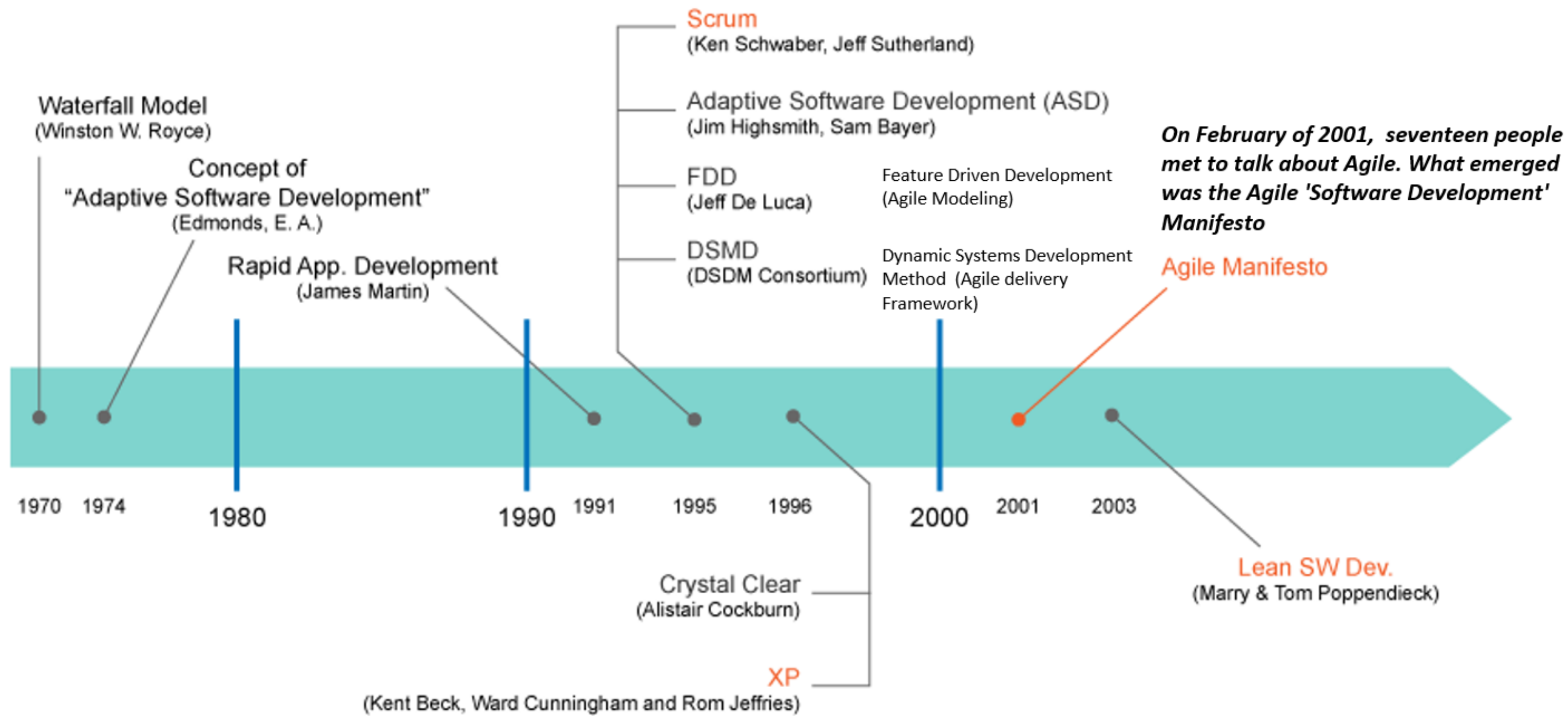
Agile History

- ❑ In the early 1990s, software development faced a crisis. Industry experts estimated that the time between a validated business need and an actual application in production was about three years.
- ❑ Within the space of three years, requirements, systems, and even entire businesses were likely to change. That meant that many projects ended up being cancelled partway through, and many of those that were completed didn't meet all the business's current needs, even if the project's original objectives were met
- ❑ In certain industries, the lag was far greater than three years. In aerospace and defense, it could be 20 or more years before a complex system went into actual use. The Space Shuttle program, which operationally launched in 1982, used information and processing technologies from the 1960s
- ❑ In 1990s, several technology leaders frustrated with these long lead times and decisions made early in a project that couldn't be changed late, began informal talks about ways to develop software more simply, without the process and documentation overhead of **Waterfall** and other popular software engineering techniques of the time

Agile Development - History

- First appeared in 1990.
- The once-common perception that agile methodologies are nothing but **controlled code-&-fix approaches**, with little or no sign of a clear-cut process, is only true of a small – albeit influential – minority.
- Essentially based on **practices of program design, coding and testing** that are believed to enhance software development **flexibility** and **productivity**.
- Most agile methodologies incorporate explicit processes, although striving to keep them as **lightweight** as possible.

Agile History Timeline

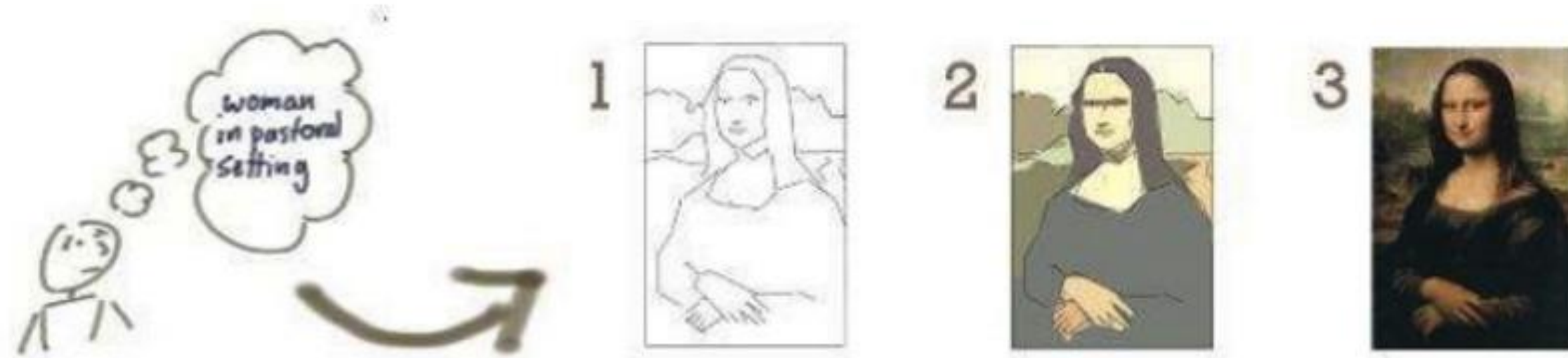


Hiroataka Takeuchi 1986 ← → 1993 Jeff Sutherland, Jeff McKenna, John Scumniotales

Evolution of Project Management

Simple Comparison	
Traditional Project Management	Modern Project Management (Agile)
Elements impacting execution of a project: <ul style="list-style-type: none"> a) Planning b) Control 	Elements impacting execution of a project: <ul style="list-style-type: none"> a) Competitive environment b) Creativity and Innovation c) Planning d) Control
Project Success is measured by having: <ul style="list-style-type: none"> a) Well defined requirements b) Approved budget c) On-Time Delivery 	Project Success is measured by: <ul style="list-style-type: none"> a) Delivery of Business Value (what does customer want) <p>Less important are:</p> <ul style="list-style-type: none"> a) Having well defined requirements b) Budget constraints c) Set Schedule <p>Today's world has a much higher level of uncertainty which makes it difficult to always start a project with well-defined requirements</p>
Requires a Project Manager	Project Manager Role is distributed among: <ul style="list-style-type: none"> a) Product Owner b) Scrum Master c) Project Team

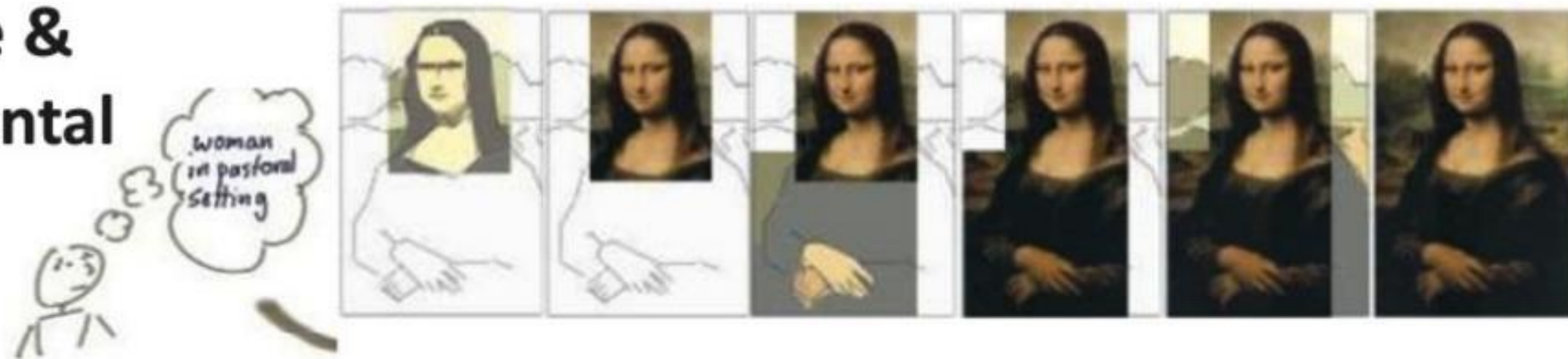
Iterative



Incremental



Iterative & Incremental



Not like this....



1



2



3



4

Like this!



1



2



3



4

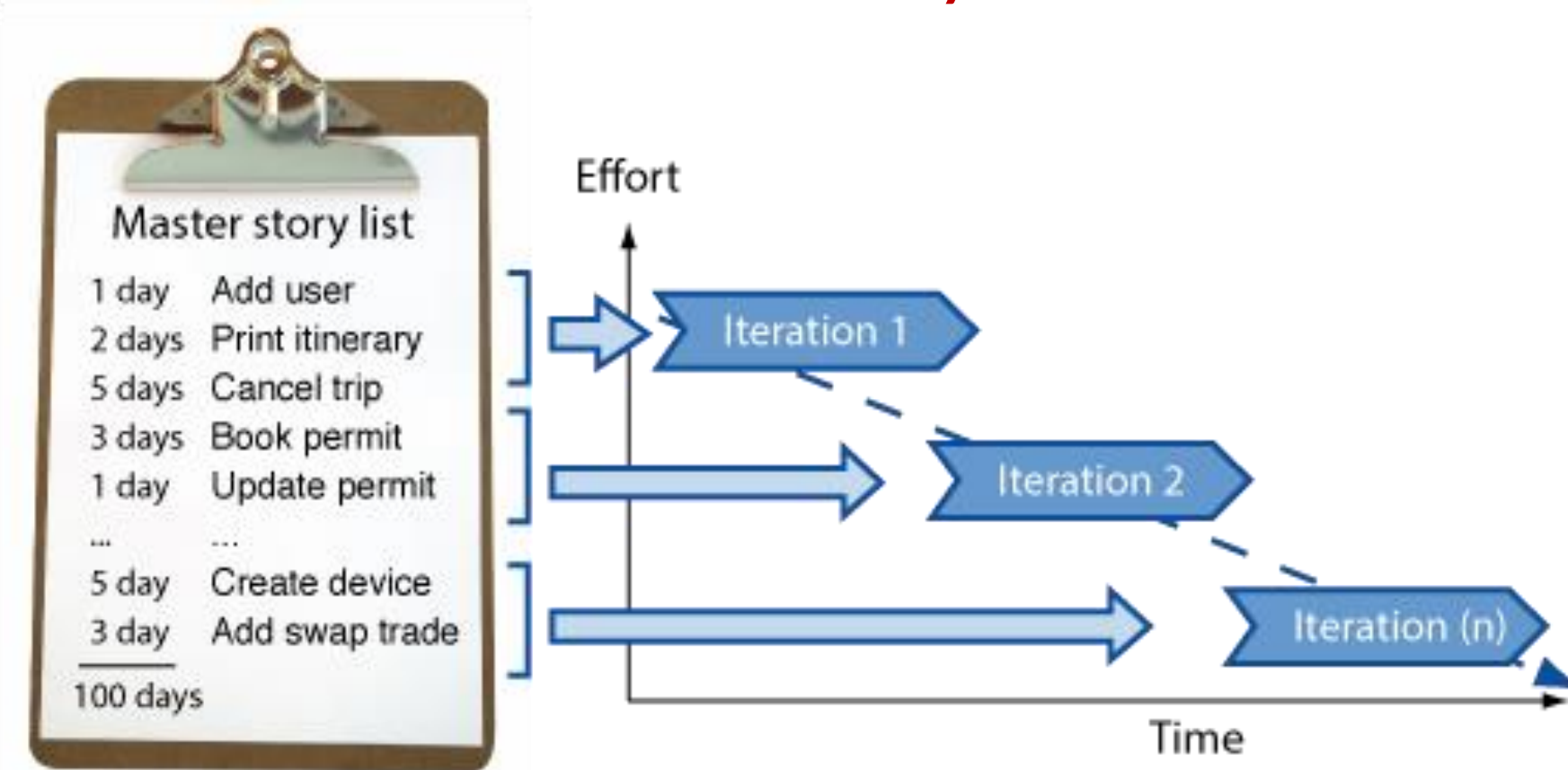


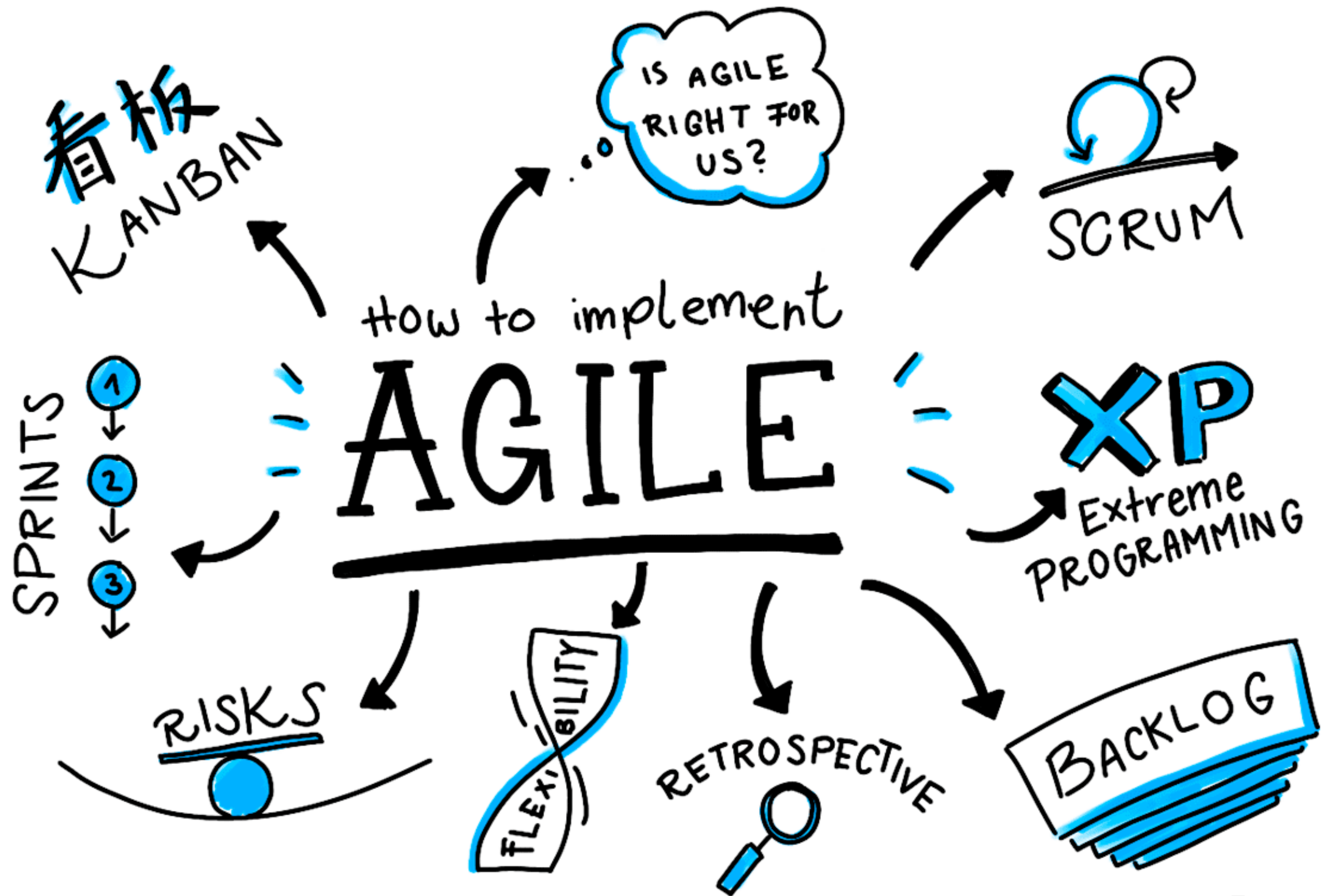
5

Henrik Kniberg

Understanding Agile

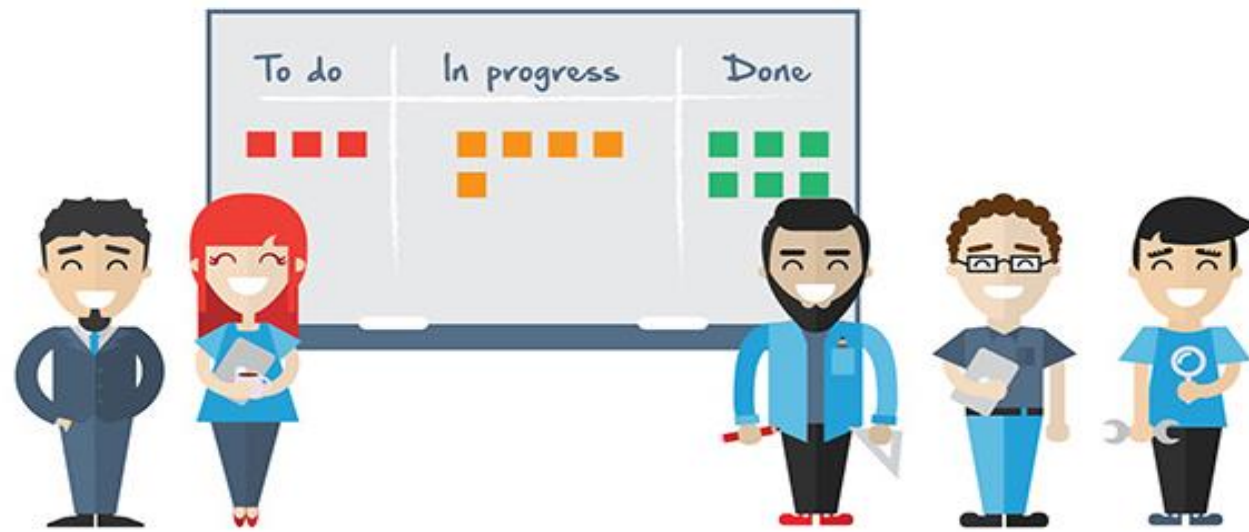
- Agile is a time boxed, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver it all at once near the end. It works by breaking projects down into little bits of user functionality called **User Stories**, prioritizing them, and then continuously delivering them in **short two week cycles called Iteration**.





*planio

Scrum, Kanban & XP Agile Methodologies



Scrum is a tool used to organize work into small, manageable pieces that can be completed by a cross-functional team within a prescribed time period (called a sprint, generally 2-4 weeks long) to plan, organize, administer, and optimize this process.

Kanban (an industrial engineer at Toyota, developed Kanban to improve manufacturing efficiency) is also a tool used to organize work for the sake of efficiency. Where Scrum limits the amount of time allowed to accomplish a particular amount of work (by means of sprints), Kanban limits the amount of work allowed in any one condition (only so many tasks can be ongoing, only so many can be on the to-do list.)

Extreme Programming (XP) is an agile software development framework that aims to produce higher quality software, while focusing on customer satisfaction by delivering what's needed when needed. Its guiding principles are: Communication, Simplicity, Feedback, Respect and Courage

Kanban vs Scrum

□ Kanban:

- Visualize Work: Uses Kanban boards to show work status.
- Continuous Flow: Work is done continuously, no fixed iterations.
- WIP Limits: Limits on work in progress to prevent overload.
- Flexibility: Allows for changes in priorities anytime.

□ Scrum:

- Sprints: Work is done in fixed iterations (usually 2-4 weeks).
- Defined Roles: Specific roles like Scrum Master and Development Team.
- Planning & Review: Planning at the start of each Sprint; review at the end.
- Focus on Iteration: Emphasizes feedback and continuous improvement.

Kanban vs Scrum



In Scrum, work is limited per iteration, called a Sprint. The development team commits to a specific number of tasks for the Sprint. While there are commitments, all items can be in progress simultaneously.	Kanban limits work in progress for each stage of the workflow. For instance, if the limit for the "In Progress" column is 5, there should not be more than 5 items in that column.
--	--

A Scrum board is owned by a Scrum Team led by a Scrum Master. The team is cross-functional, with all the skills needed for Sprint's tasks.	Kanban boards are not bound to a specific team; they primarily focus on workflow management.
--	--

In Scrum, the entire team works collectively on each task.	Team members are responsible for specific steps in the task flow (e.g., coding, testing). Slack resources in Kanban help address bottlenecks.
--	---

The Product Owner can't edit the board once the team has committed to several items for the Sprint.	A Product Owner can edit a Kanban board. Kanban allows different team roles like Service Request Manager as an alternative to the Product Owner.
---	--

New items cannot be added to the board during a Sprint in Scrum.	Updates on a Kanban board can happen without timeframes as long as work-in-progress limits are maintained.
--	--

Scrum vs XP



- ❑ Scrum teams typically work in iterations (called sprints) that are from two weeks to one month long. XP teams typically work in iterations that are one or two weeks long.
- ❑ Scrum teams do not allow changes into their sprints. XP teams welcome changes as long as work has not started
- ❑ Extreme Programming teams work in a strict priority order. Scrum teams have flexibility to choose what prioritized features to work on
- ❑ Scrum doesn't prescribe any engineering practices. XP does (things like test-driven development, focus on automated testing, pair programming, simple design, refactoring, and so on)

Major Agile Methodologies

- ❑ DSDM – Dynamic Systems Development Method (1995)
- ❑ Scrum (1995)
- ❑ XP – Extreme Programming (1996)
- ❑ ASD – Adaptive Software Development (1997)
- ❑ Crystal Family: Orange, Orange Web, Clear (1998, 2001, 2004)
- ❑ FDD – Feature-Driven Development (1999)
- ❑ AUP – Agile Unified Process (2005)

Definition of the term “manifesto”

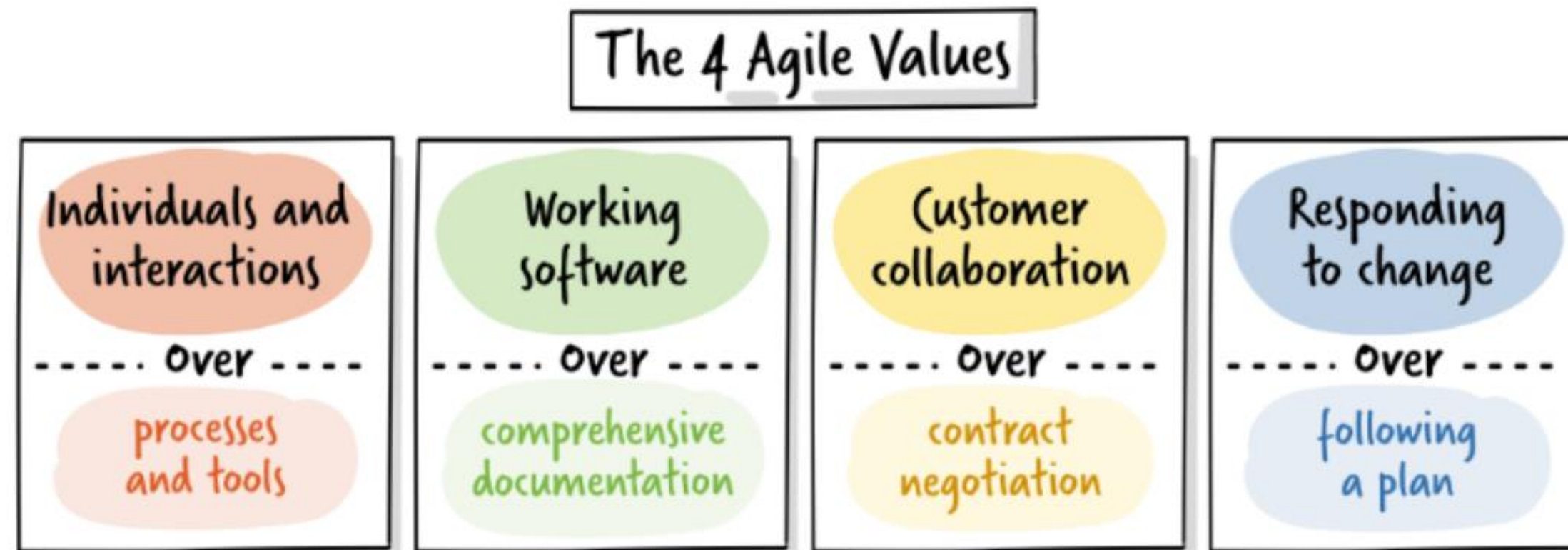
- A public declaration of principles, policies, or intentions, especially of a political nature.

2001: The Agile Manifesto is Created

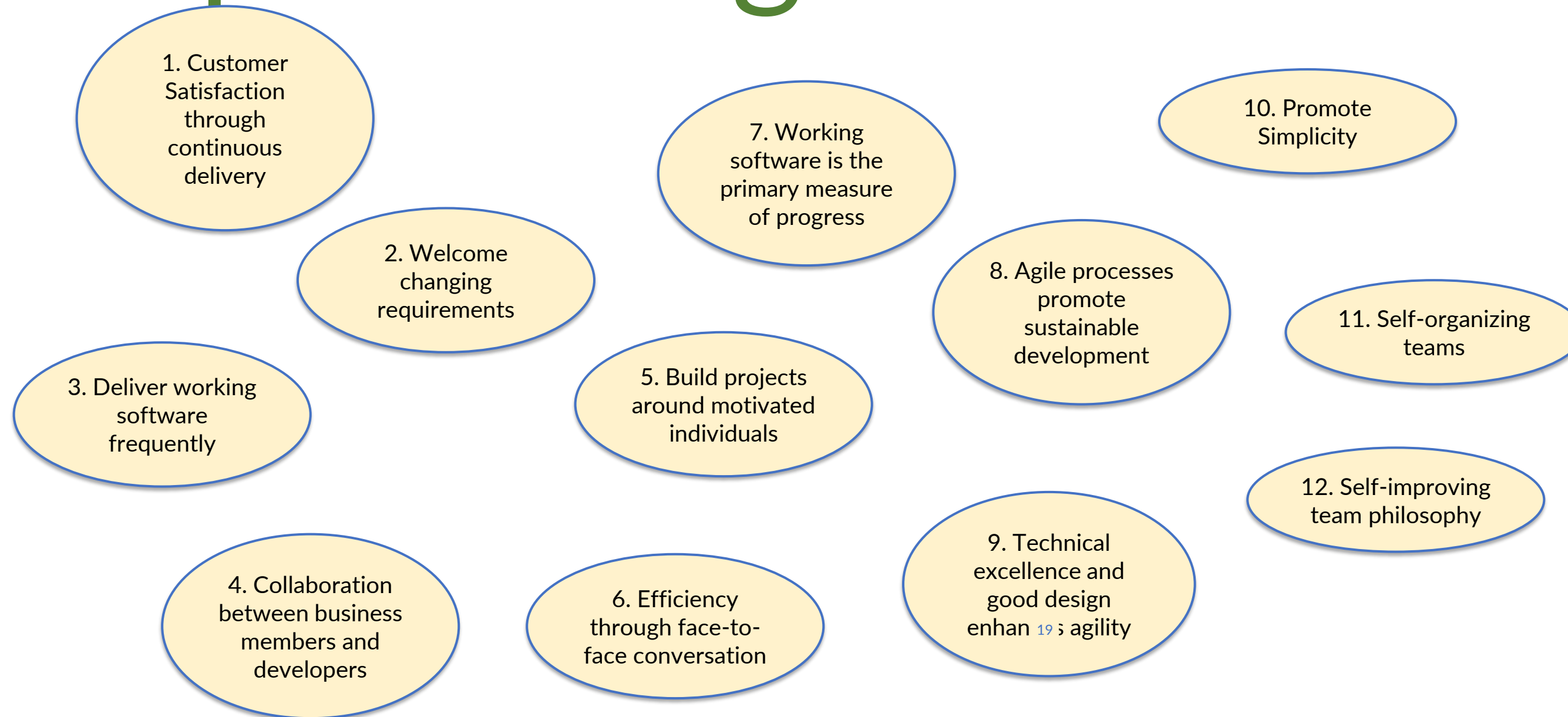
The Agile Manifesto was created and signed by 17 evangelists that observed the increasing need for an alternative to documentation-driven and heavyweight software development processes. It consists of 4 values and 12 principles and continues to be the basis for all agile software development until today.

Four Values of The 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



Principles of Agile Manifesto



Principles of 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.

Principles of Agile Manifesto

4. 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.

Principles of Agile Manifesto

- 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.

Principles of Agile Manifesto

- 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.

Source: <http://agilemanifesto.org>

Project Schedule / Cost Estimation Techniques

❑ Waterfall approach:

Bottom-Up: Detail out all requirements and estimate each task to complete those requirements in hours/days, then use this data to develop the project schedule. In the software industry, the use of the bottom-up method has severe drawbacks due to today's speed of change. *Speed of change* means that the speed of new development tools and the speed of access to new knowledge is so great that any delay in delivery leaves one open to competitive alternatives and in danger of delivering an obsolete product

❑ Agile approach:

Top-Down: The top-down method addresses this key issue, by using the information currently available to provide gross-level estimates. Rolling-wave planning is then used to incorporate new information as it's learned, further refining estimates and iteratively elaborating with more detail as the project progresses. This method of learning just enough to get started, with a plan to incorporate more knowledge as work outputs evolve, allows the project team to react quickly to adversity and changing market demand

How to estimate project schedule in Agile?



- ❑ Determine ***Point Value*** for each item to be worked on. The most popular technique of gross level estimation is ***Planning Poker***, using Fibonacci sequence to assign a point value to a feature or item
- ❑ Determine ***Team Velocity***. A team's average velocity is used in forecasting a long-term schedule. Average velocity is calculated by summing and averaging the velocity measurements from the team's last three iterations
- ❑ The team's average velocity number is used to calculate the most likely scenario, while velocity numbers from the team's worst-performing iterations are used to calculate the most ***pessimistic forecast*** completion date. Using velocity from iterations where the team was able to complete more than expected provides the most ***optimistic forecast***

How to estimate project cost in Agile?



- A simple formula is used to determine the cost per point:

Σ (loaded team salaries for period n) / points completed in period n

A team whose total loaded salaries are \$240,000 over six weeks, and completed 60 points of work in those three iterations, would have a cost per point of \$4,000

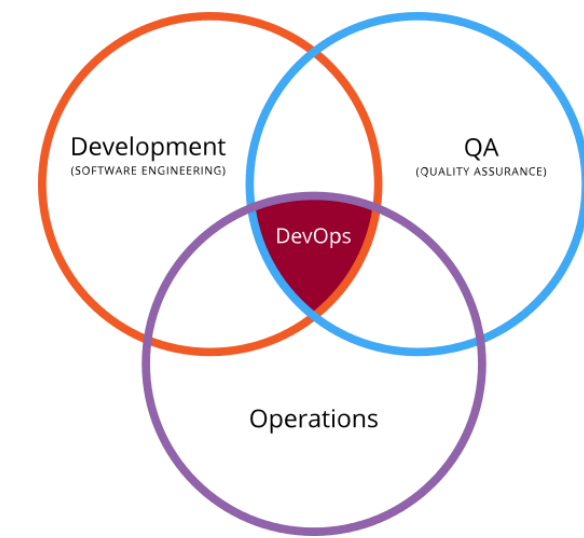
- Use the following formula to determine budget:

(Cost per point x total point value of items to be completed) + other expenses = forecast budget

Budget estimates are based on what we know today

In the example used, budget estimate is for the first release and not the entire project. The team could apply an additional 20% for the second release and an additional 5% for the last release, based on past experience

What is DevOps?



- ❑ We are now on Agile bandwagon (hooray)! Development teams are working on the software in short sprints lasting not more than two weeks.
- ❑ Having such a short release cycle helped the development team work on client feedback and incorporate it along with bug fixes in the next release.
- ❑ While this Agile SCRUM approach brought agility to development, it was lost on Operations which did not come up to speed with Agile practices. Lack of collaboration between Developers and Operations Engineers still slowed down the development process and releases.

Venn Diagram

- ❑ **DevOps Methodology** was born out of the need for better collaboration and faster delivery. DevOps enables continuous software delivery with less complex problems to fix and faster resolution of problems.
- ❑ In summary: **Agile** is a set of values and principles about how to develop software. If you have ideas and you want to turn those ideas into working software, you can use the Agile values and principles as a way to do that. But, that software might only be working on a developer's laptop or in a test environment. You want a way to quickly, easily and at will move that software into production infrastructure, in a safe and simple way. To do that you need **DevOps** tools and techniques.

Does Agile work for infrastructure projects?



- ❑ Agile works very well in projects where more is unknown than known. This is common in **software development** projects. In software development end users kind of know what they want. The requirements evolve over time. For example: end user thinks they need some kind of a shape. You start with something like circle. After an iteration the requirement might change from circle to square. After another iteration it might change from square to square filled with some color. The requirements evolve. Agile methodology has framework to support "**unknown**" throughout the project life cycle.
- ❑ In contrast **infrastructure** projects have very well defined requirements (very few unknowns), strict dependency between tasks, mainly configuration driven, some tasks are done only during permitted windows (weekends or after business hours). For example ***storage expansion*** tasks are comprised of procurement, cabling, network and storage tasks. These tasks have dependency and follow one after another. Projects like this are well suited for waterfall methodology. But infrastructure projects which have "**unknowns**" and similar to software development (BaaS, IaaS, PaaS) can greatly benefit from agile methodology

When Agile Fails?

- ❑ System Criticality
 - Comfort (C)
 - Discretionary Money (D)
 - Essential Money (E)
 - Life (L)

Cynefin Framework

□ A decision framework

