
# Agile Software Development

Dario Campagna


## Utah, February 2001

A group of 17 process experts met to discuss the growing field of what used to be called lightweight methods.

# Agile, why?

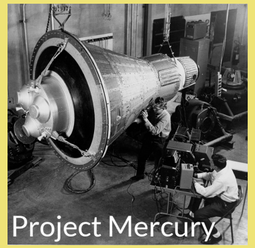
We are uncovering better ways of developing software by doing it and helping others do it.

**We are uncovering** better ways of developing software by doing it and helping others do it.

# Iterative and Incremental Development

W. **Shewhart** “plan-do-study-act” (**PDSA**) cycles at Bell Labs

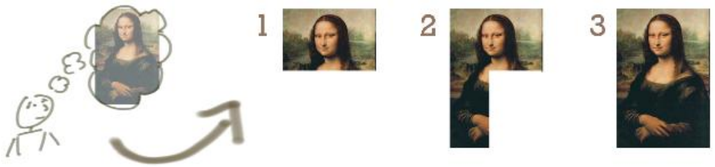
W. E. **Deming** vigorously promoting PDSA

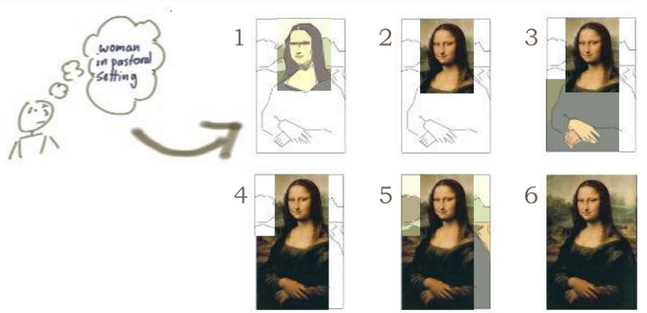

1930 1940 1950 1960 1970

# Iterative and Incremental Development


**Iterate** to evaluate and make changes to what you've already made

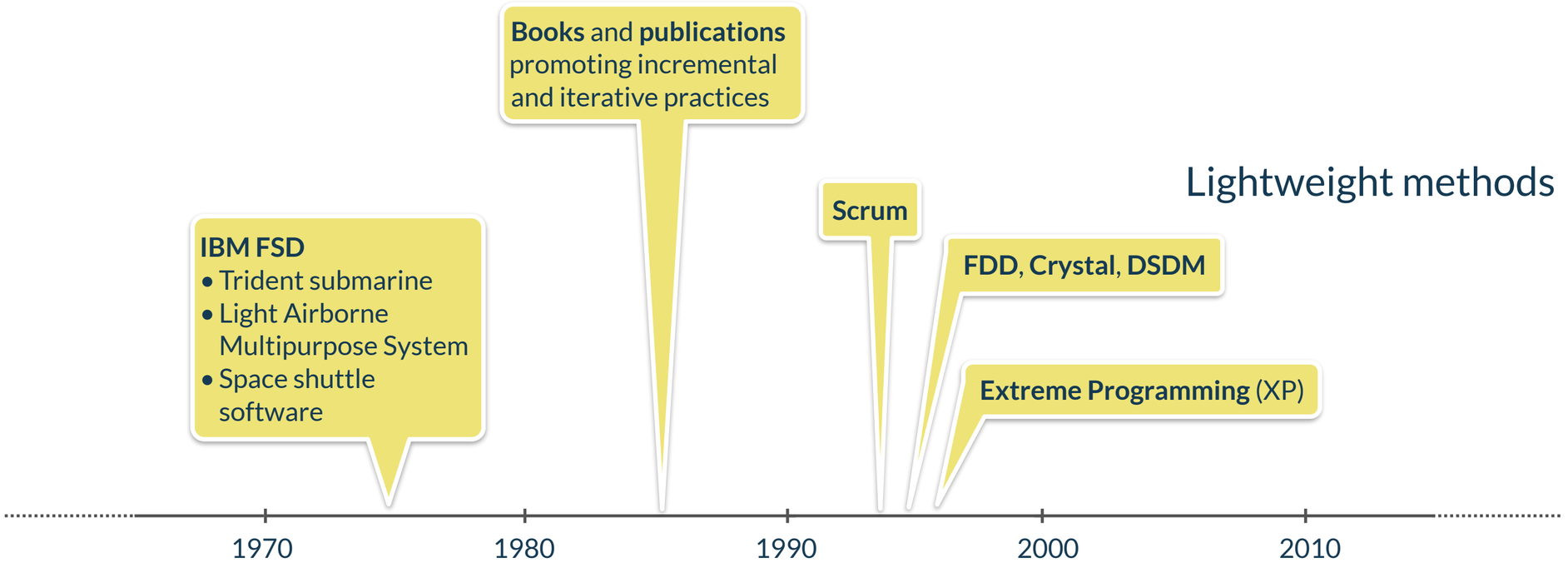



**Increment** to make additions

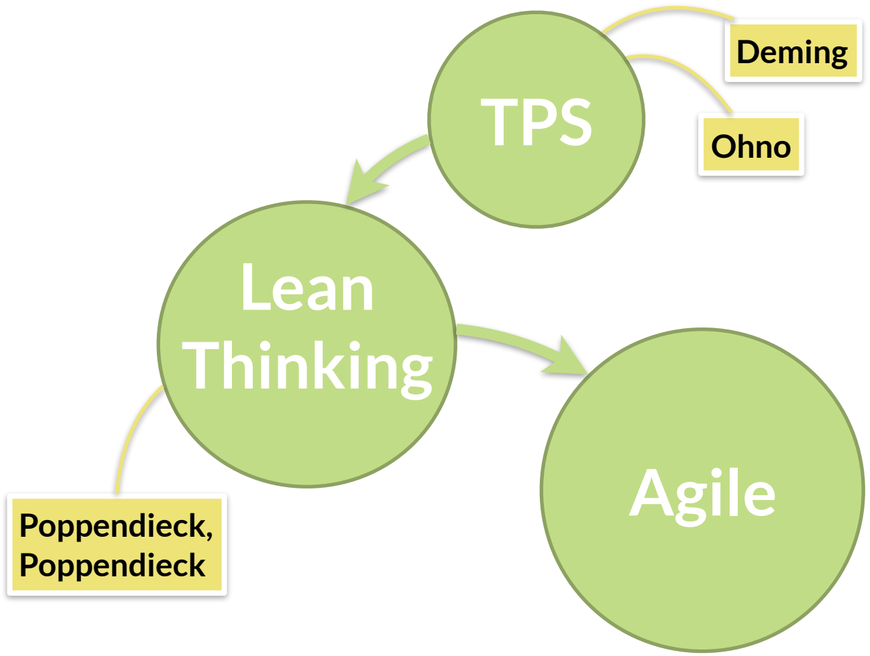



The two tactics can be **conjoined**

# From IBM FSD to Agile Methods



# Toyota Production System and Lean Thinking



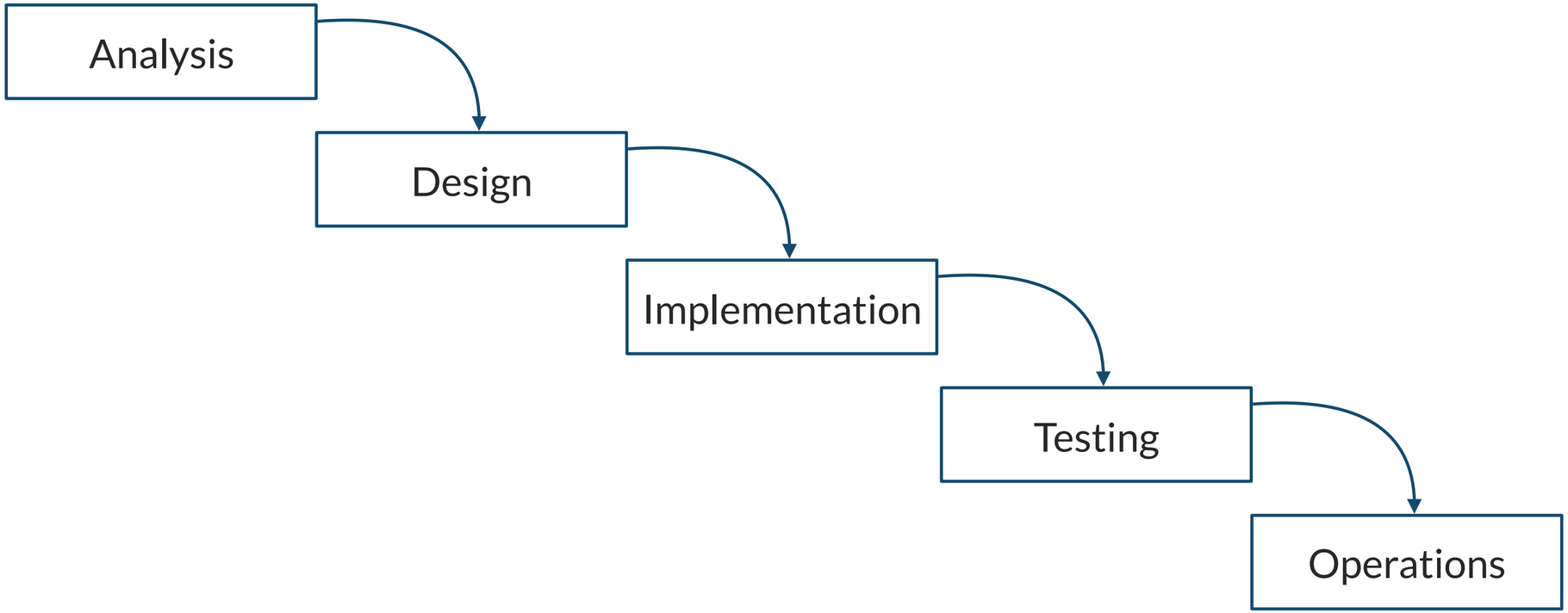

TPS is the precursor to the Toyota Way, a.k.a. Lean Thinking

Agile methods have been influenced by Lean Thinking

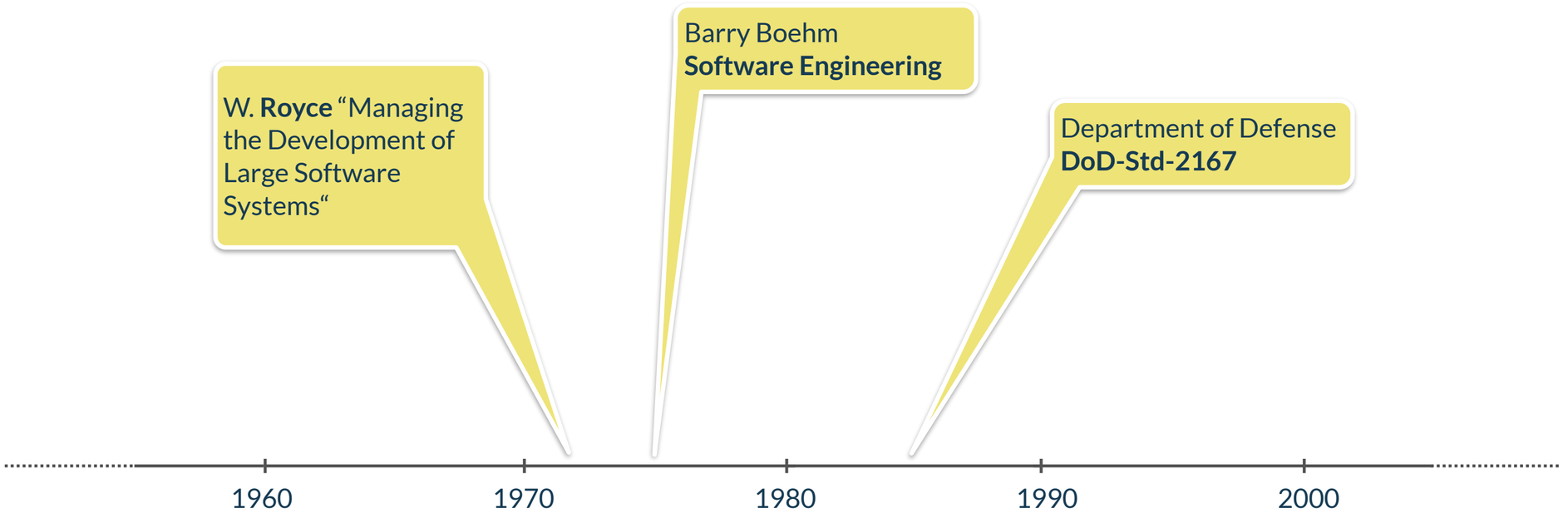
Correspondence and complementary qualities of lean to agile methods

We are uncovering **better ways of developing software** by doing it and helping others do it.

# Strict, document-driven, single-pass waterfall model



# Strict, document-driven, single-pass waterfall model



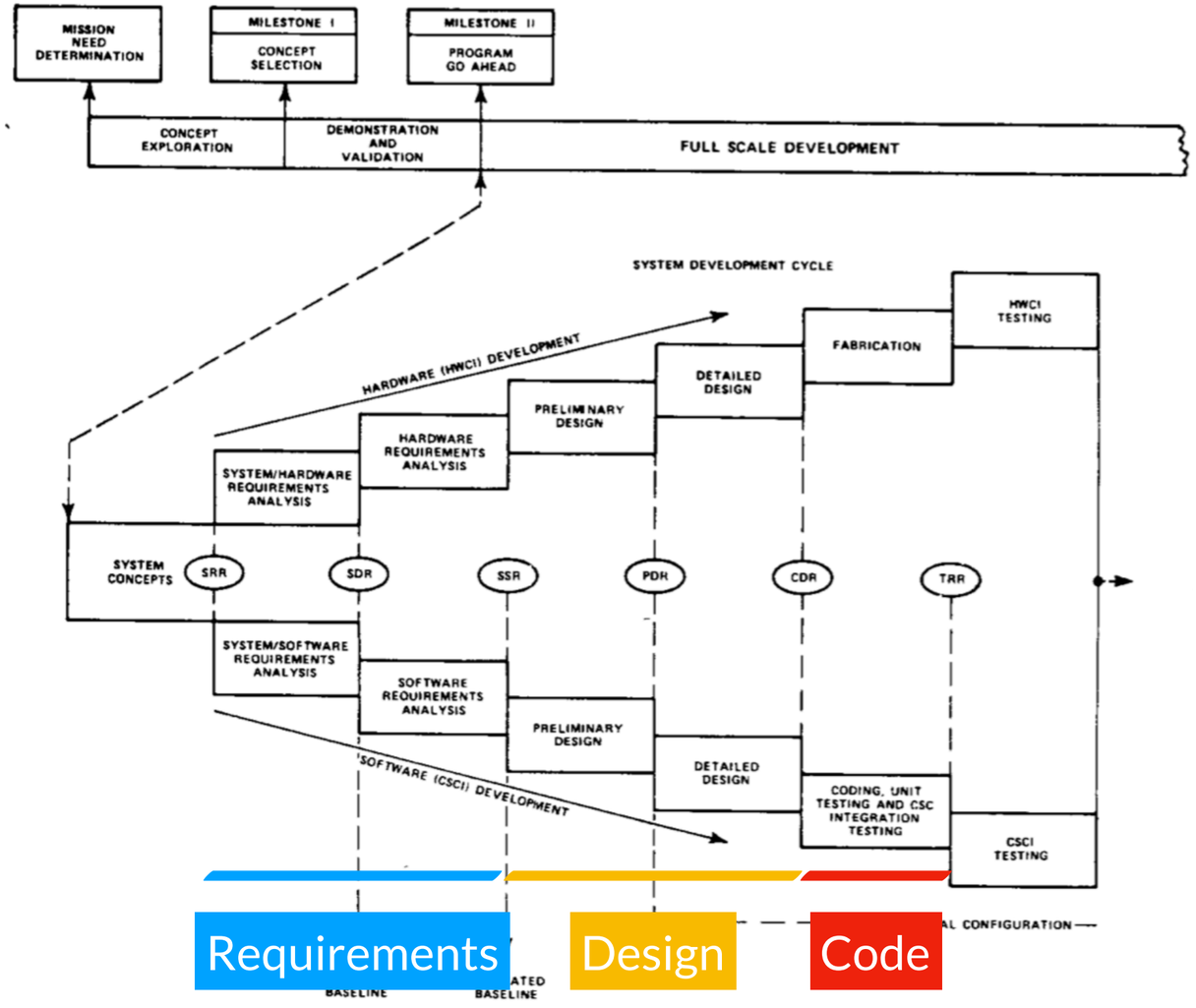



**FIGURE 1.** System development cycle within the system life cycle.

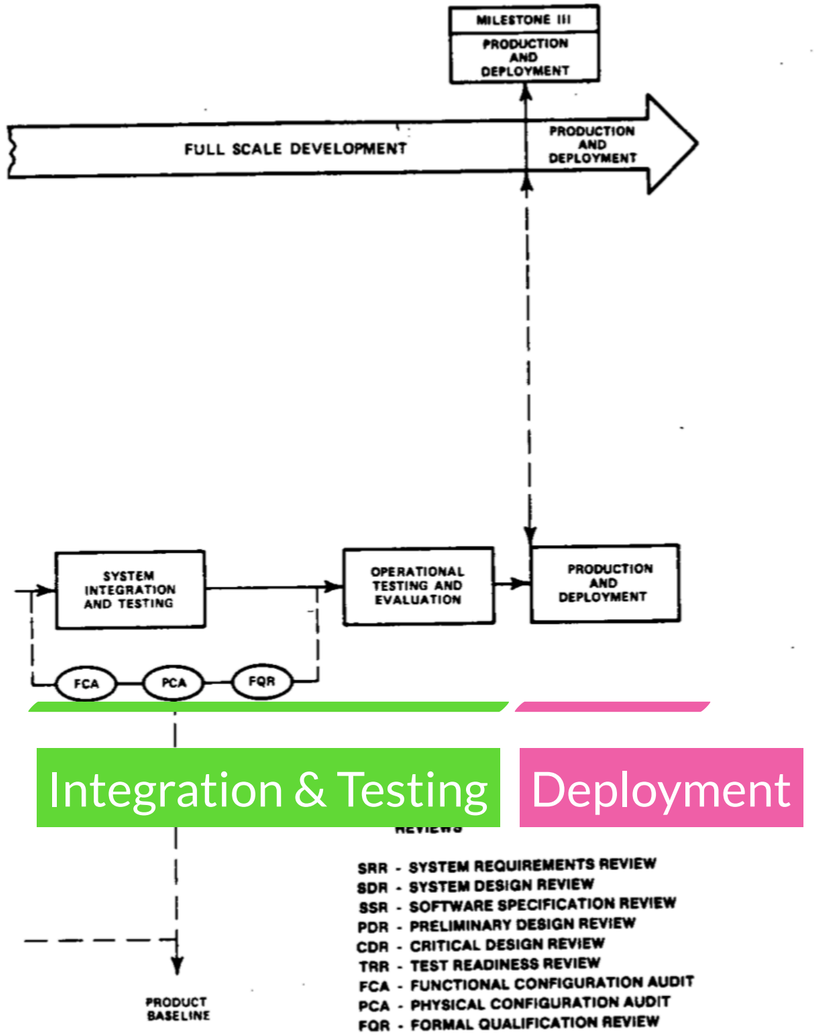
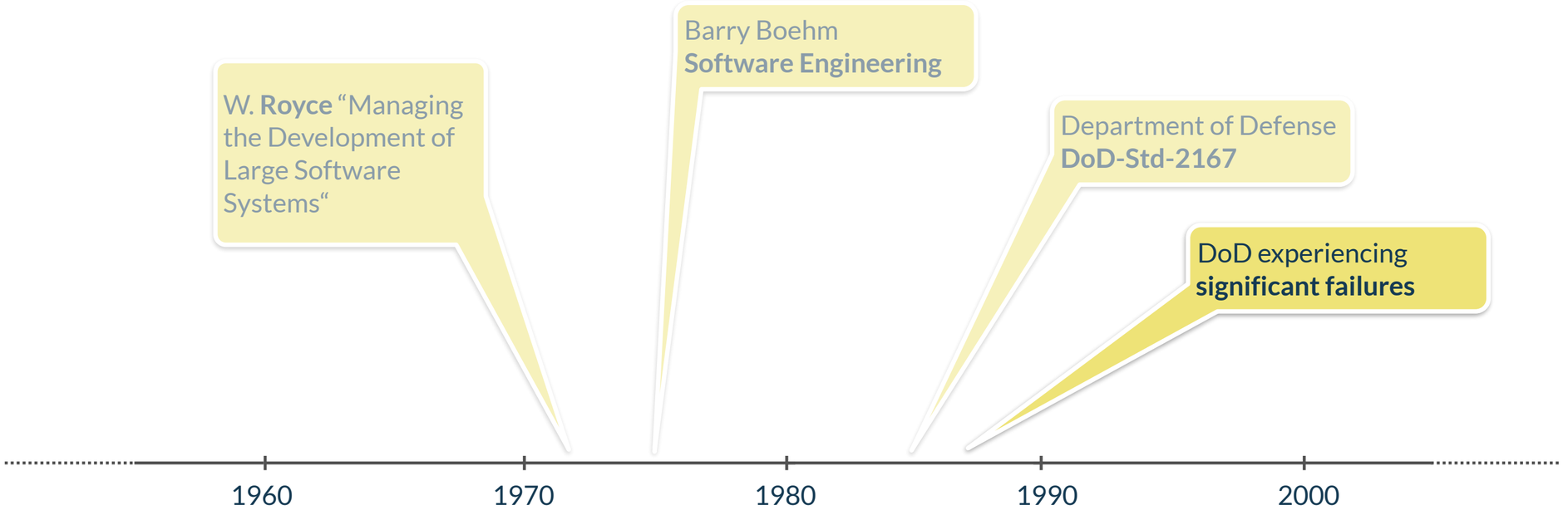



**FIGURE 1.** System development cycle within the system life cycle. (continued)

# Strict, document-driven, single-pass waterfall model



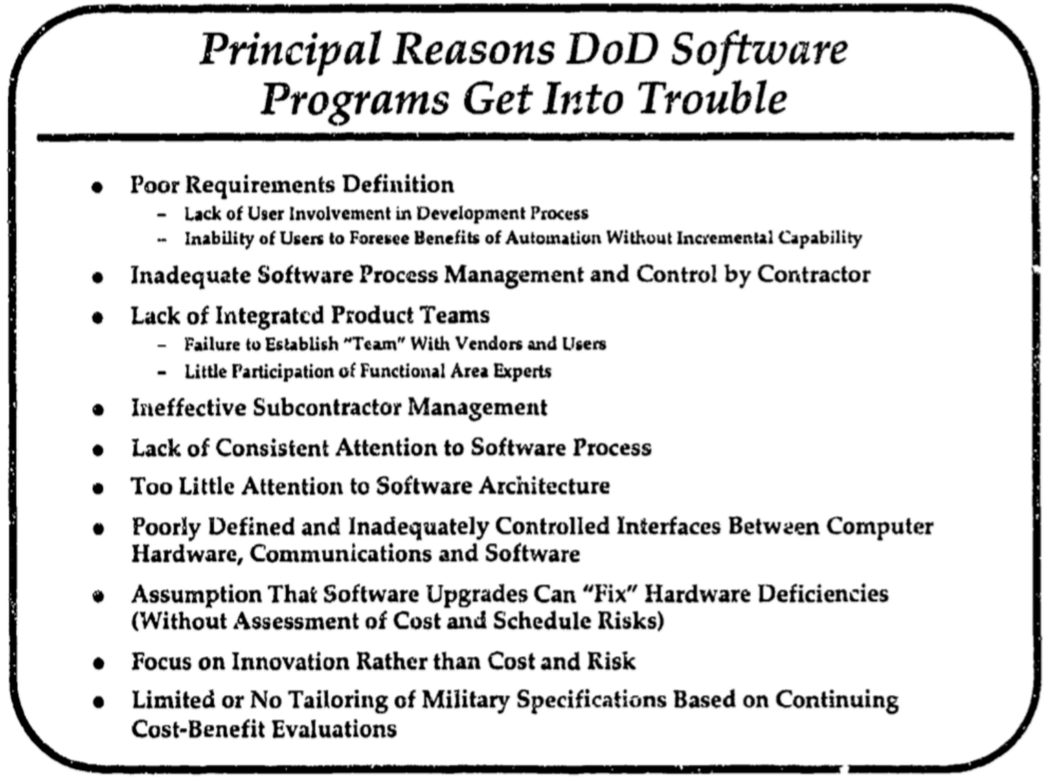
# Why did projects fail?

1987 report of the Defense Science Board Task Force on Military Software.

> Directive 5000.29 not only does not encourage this best modern practice, it essentially forbids it. We recommend that it be revised immediately to mandate and facilitate early prototyping before the baseline specifications are established (Rec. #23).
>
> DoD-STD-2167 likewise needs a radical overhaul to reflect best modern practice. Draft DoD-STD-2167A is a step, but it does not go nearly far enough. As drafted, it continues to reinforce exactly the document-driven, specify-then-build approach that lies at the heart of so many DoD software problems.

# Why did projects fail?

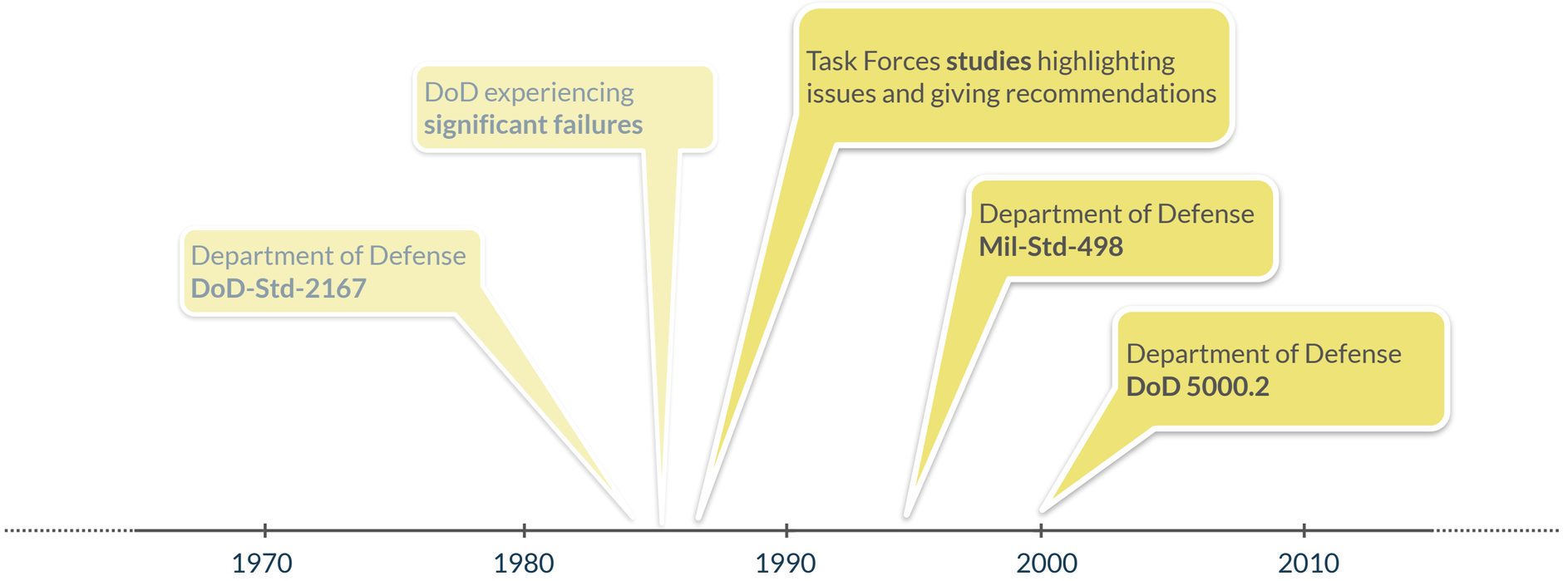
1994 report of the Defense Science Board Task Force on Acquiring Defense Software Commercially.



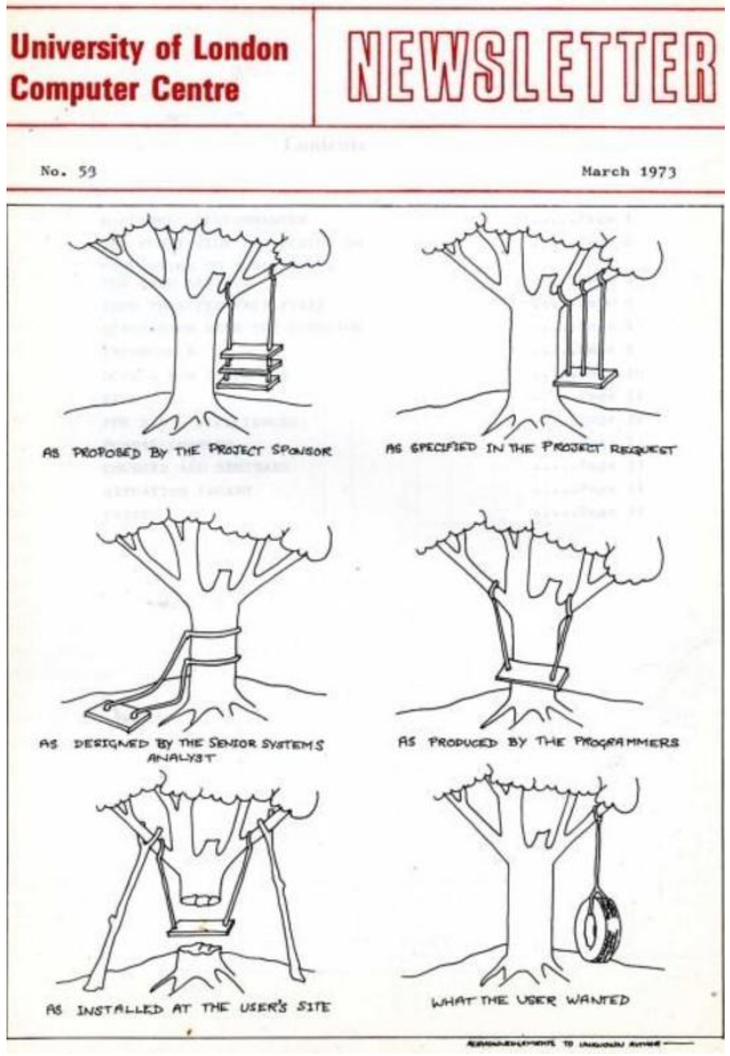
*Principal Reasons DoD Software Programs Get Into Trouble*

- Poor Requirements Definition
  - Lack of User Involvement in Development Process
  - Inability of Users to Foresee Benefits of Automation Without Incremental Capability
- Inadequate Software Process Management and Control by Contractor
- Lack of Integrated Product Teams
  - Failure to Establish "Team" With Vendors and Users
  - Little Participation of Functional Area Experts
- Ineffective Subcontractor Management
- Lack of Consistent Attention to Software Process
- Too Little Attention to Software Architecture
- Poorly Defined and Inadequately Controlled Interfaces Between Computer Hardware, Communications and Software
- Assumption That Software Upgrades Can "Fix" Hardware Deficiencies (Without Assessment of Cost and Schedule Risks)
- Focus on Innovation Rather than Cost and Risk
- Limited or No Tailoring of Military Specifications Based on Continuing Cost-Benefit Evaluations

# Strict, document-driven, single-pass waterfall model



# The Problem with Requirements


University of London Computer Centre

NEWSLETTER

No. 53 March 1973

AS PROPOSED BY THE PROJECT SPONSOR

AS SPECIFIED IN THE PROJECT REQUEST

AS DESIGNED BY THE SENIOR SYSTEMS ANALYST

AS PRODUCED BY THE PROGRAMMERS

AS INSTALLED AT THE USER'S SITE

WHAT THE USER WANTED

Newsletter of the University of London Computer Center

March 1973

# Management Theories

## Scientific Management

- Best practices should be pushed throughout the organization
- Planning and improvement work separated from normal work

## General and Industrial Management

Responsibilities of managers

- Planning
- Organizing
- Coordinating
- Commanding
- Controlling

# Charlie and Jane

# Characteristics of Agile

# Adaptability

## Adaptability as a driver

⬆ Agility/Adaptability

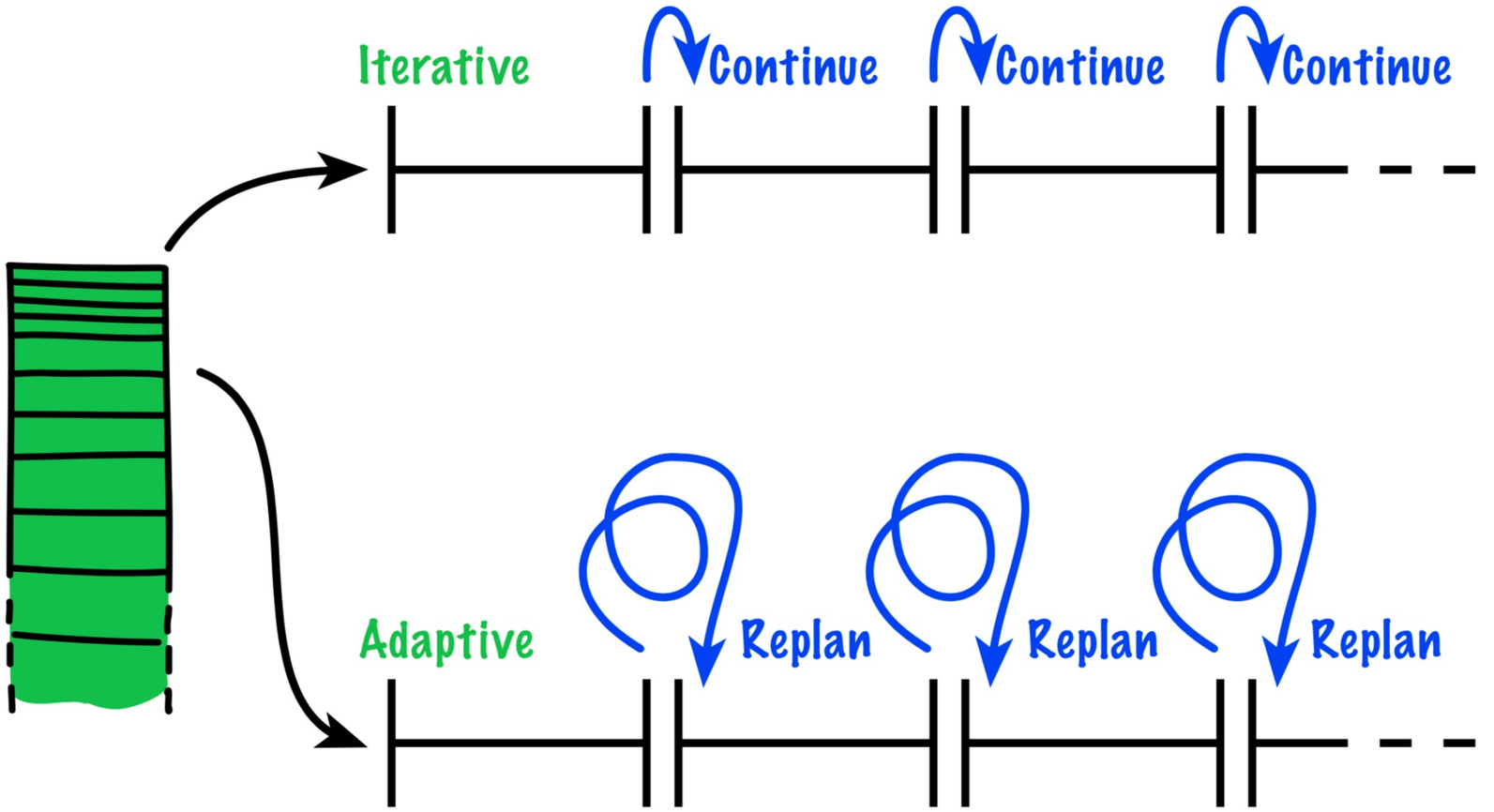
⬆ Value

## Common misunderstanding

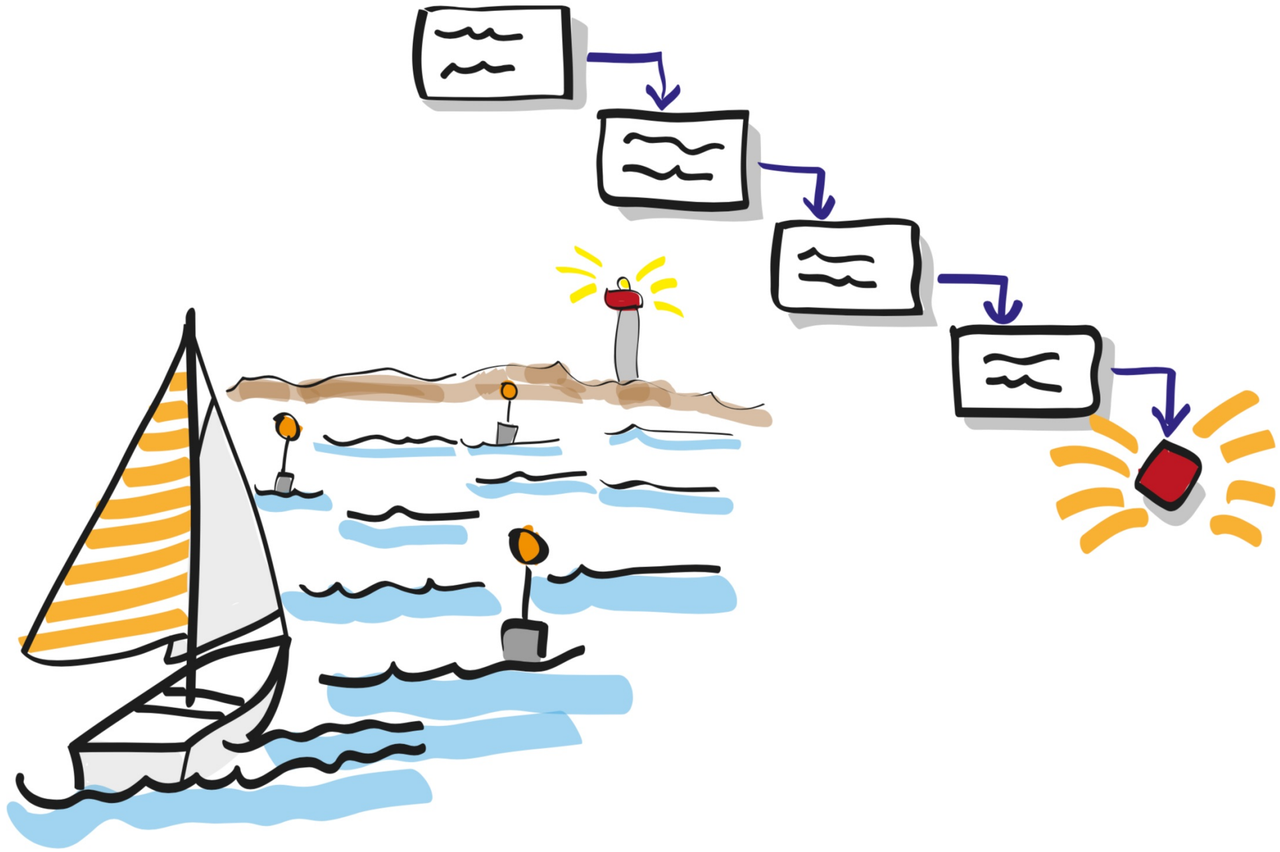
Agility ≠ Fast

Agility ≠ Cheap

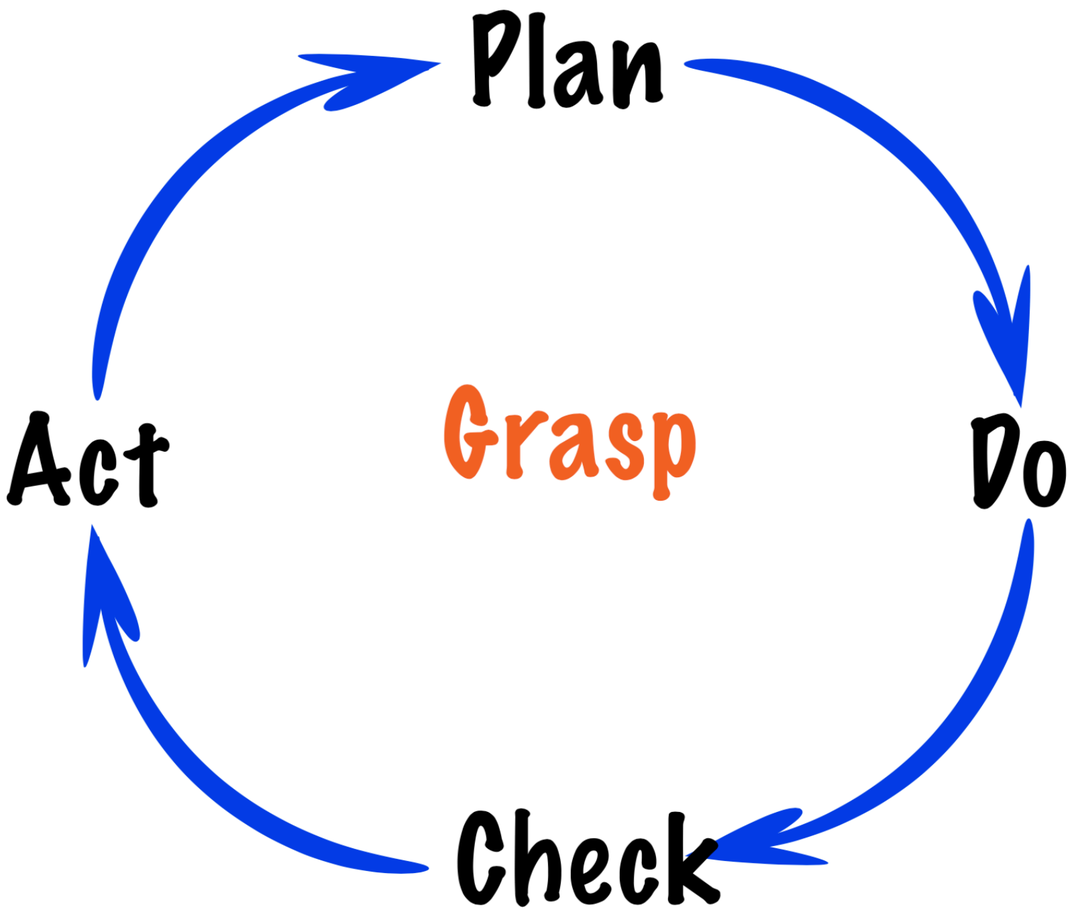
# Adaptive Vs Iterative



# Empirical Vs Defined Process



# Continuous Improvement



# Manifesto for Agile Software Development

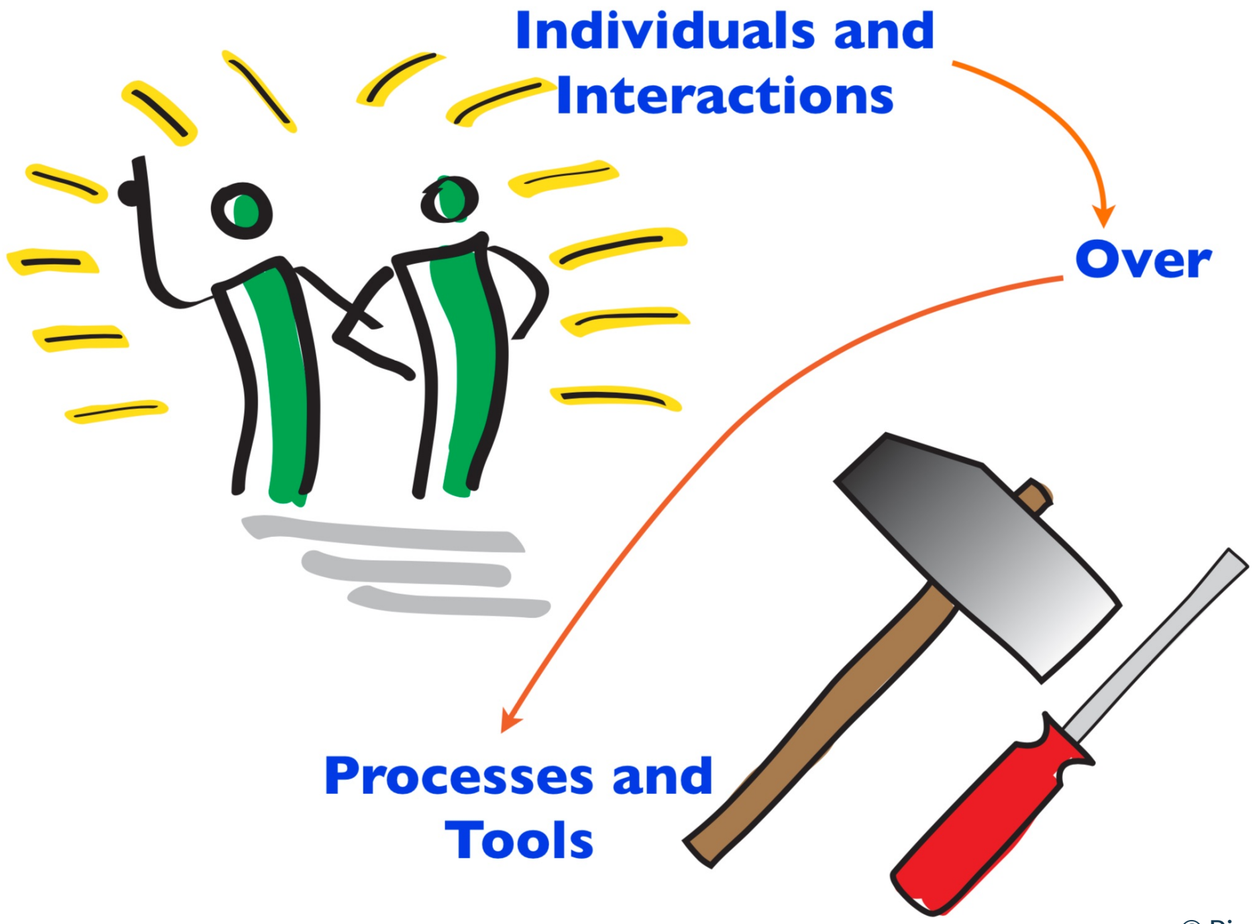
We are uncovering better ways of developing software by doing it and helping others do it.

Through this work we have come to value:

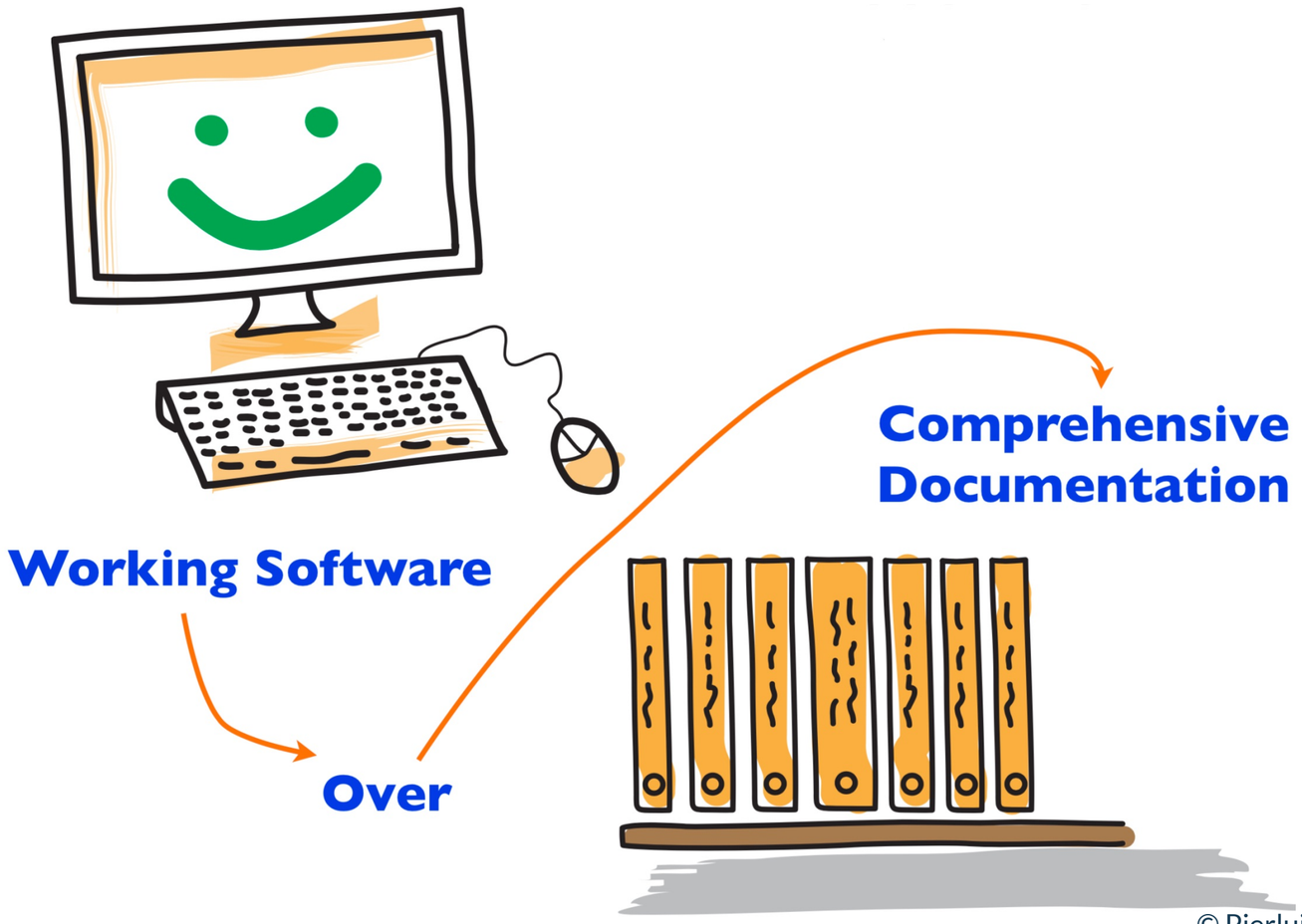
[X over Y]

That is, while there is value in the items on the right, we value the items on the left more.

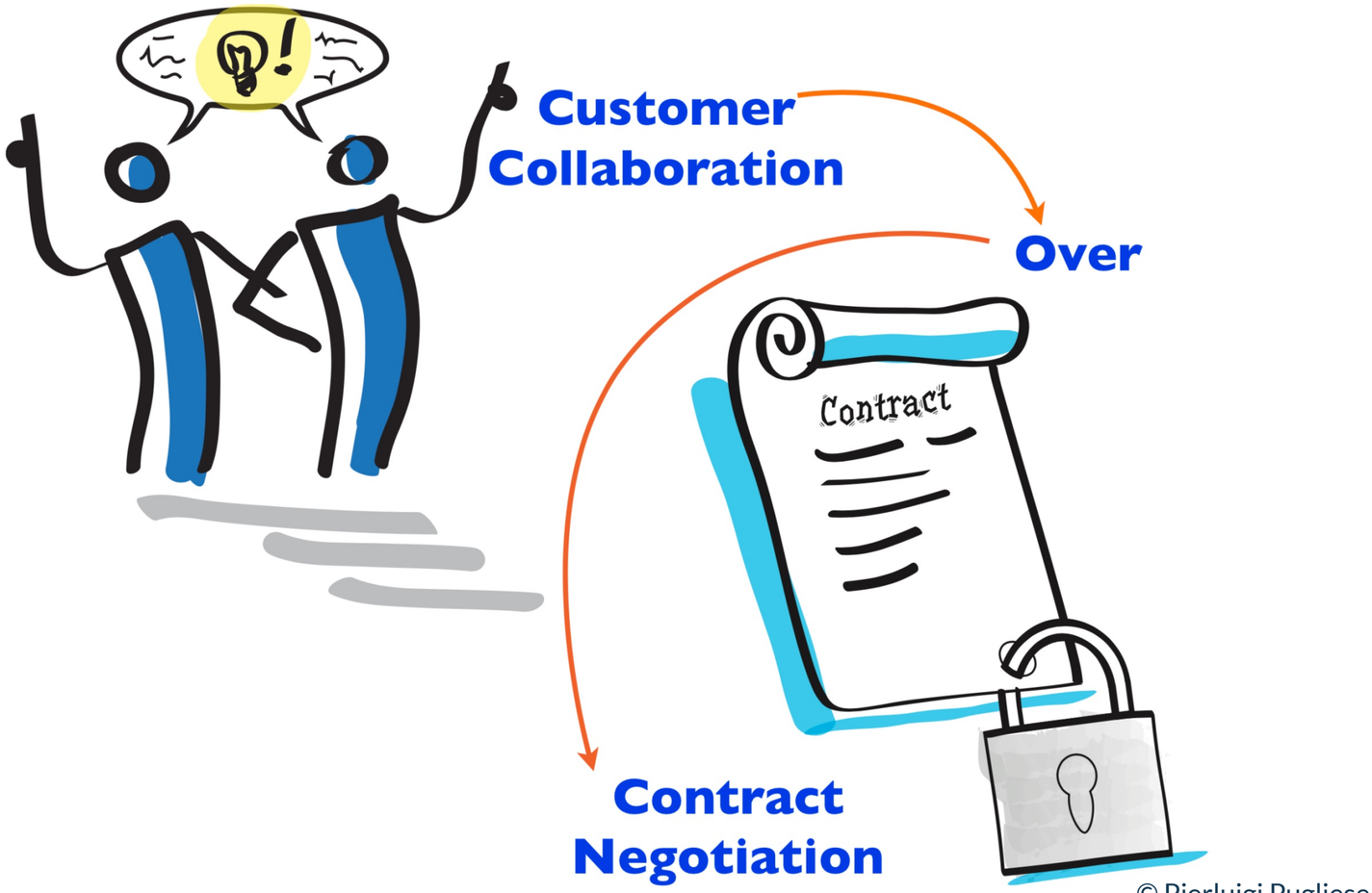
# We value…



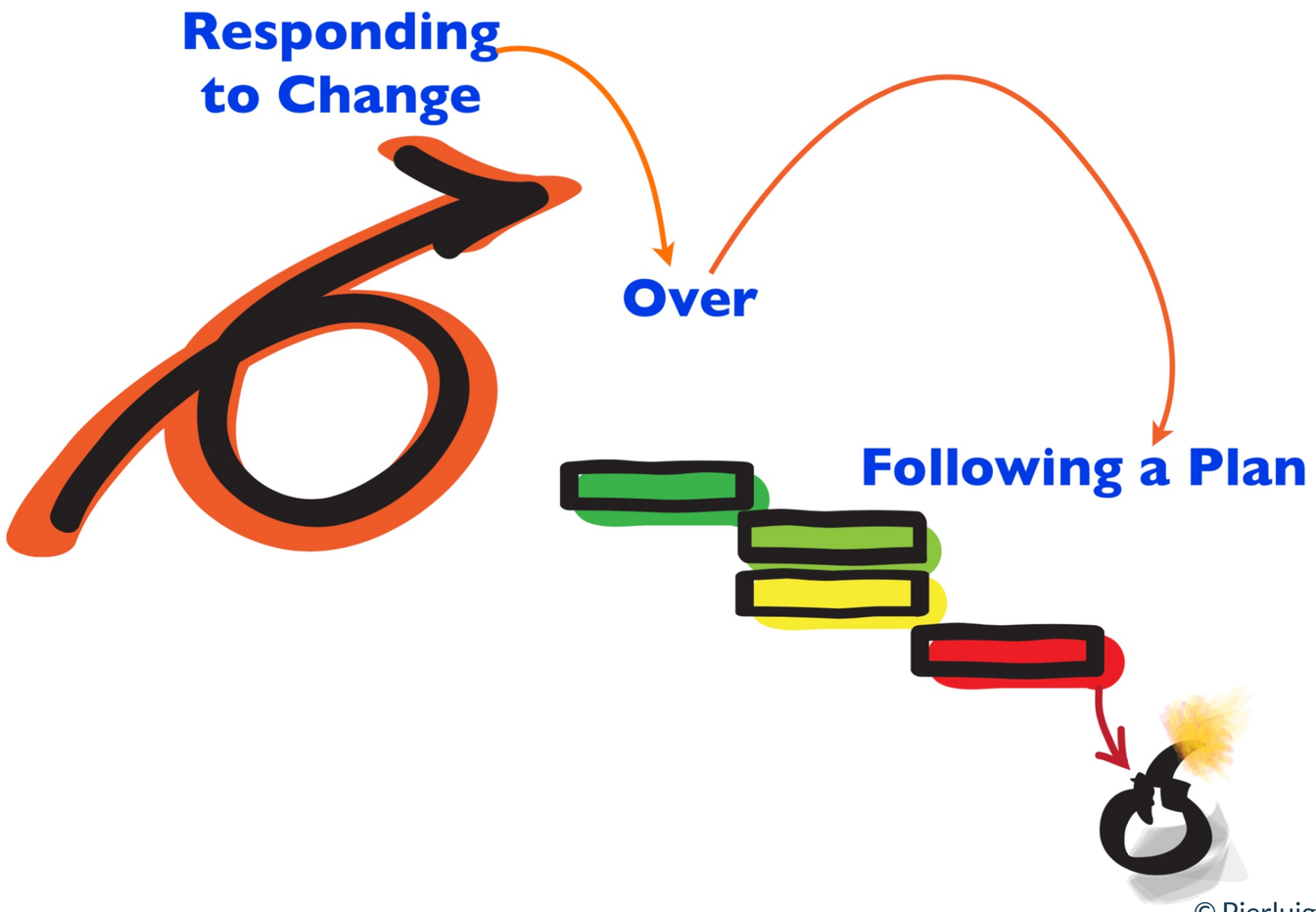
# We value…



# We value…



# We value…



# Agile Manifesto Principles

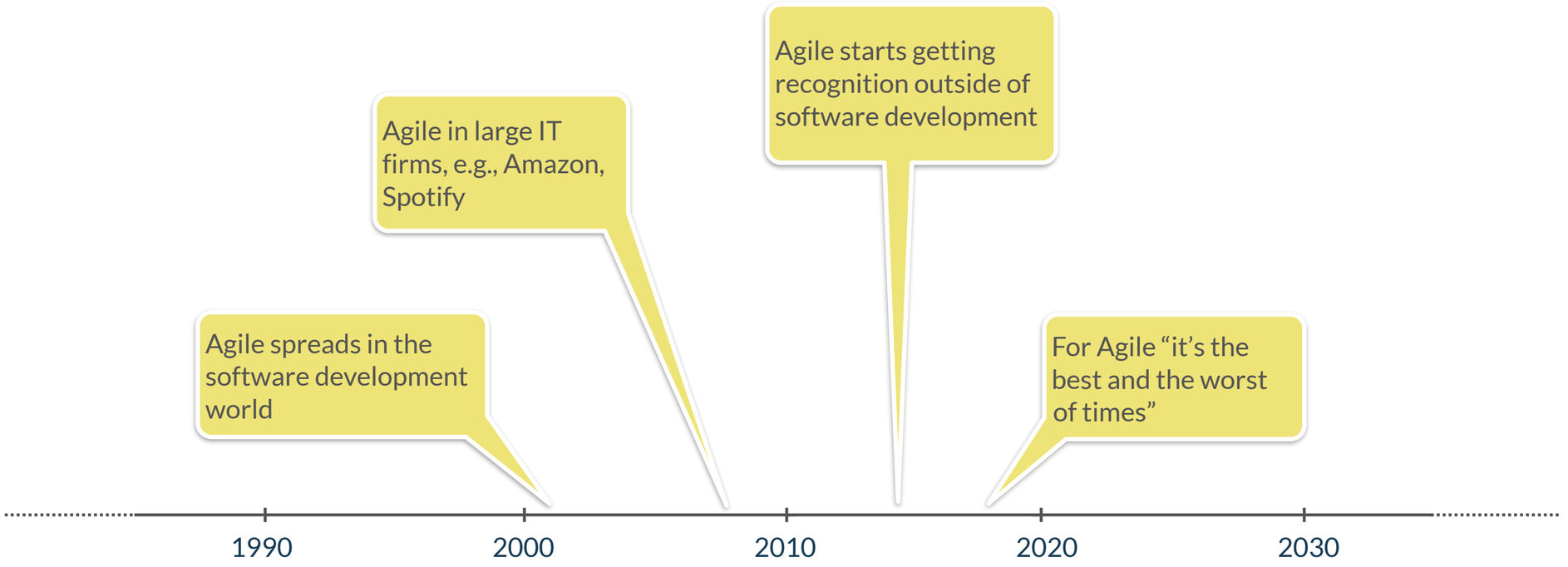
| | | | |
|---|---|---|---|
| 1 | Our highest priority is to satisfy the customer through early and continuous delivery of valuable software. | 4 | Business people and developers must work together daily throughout the project. |
| 2 | Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage. | 5 | Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done. |
| 3 | Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale. | 6 | The most efficient and effective method of conveying information to and within a development team is face-to-face conversation. |

# Agile Manifesto Principles

| | | | |
|---|---|---|---|
| 7 | Working software is the primary measure of progress. | 10 | Simplicity – the art of maximizing the amount of work not done – is essential. |
| 8 | Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely. | 11 | The best architectures, requirements, and designs emerge from self-organizing teams. |
| 9 | Continuous attention to technical excellence and good design enhances agility. | 12 | At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly. |

# What happened after 2001?

# During the last 18 years



# Agile is enjoying both

## The best of times

- Ways to deliver instant, intimate, incremental, risk-free value at scale
- Spreading from IT Department to all parts, and all kinds, of organizations
- Ridicule of Agile turned to envy

## The worst of times

- Agile implemented as a superficial patch on traditional management
- Some consultant and coaches are selling “get Agile quick” schemes
- Huge amount of “fake Agile” going on
- Risk that Agile is being dumbed down as to become a shadow of the real thing

# Quick introduction to Scrum

# Takeuchi and Nonaka

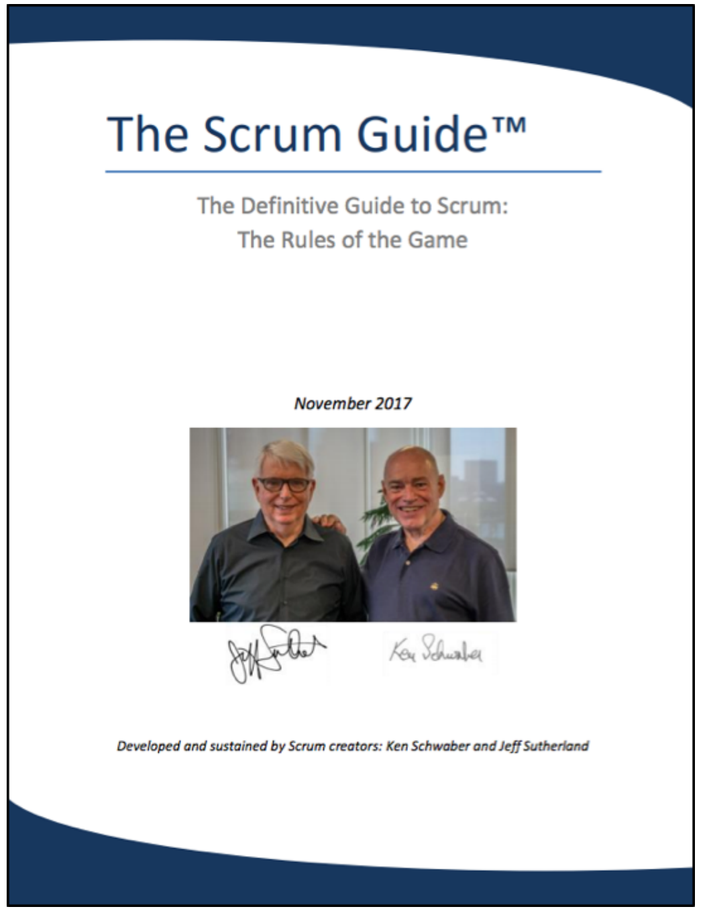
“The New New Product Development Game”

- Used the term Scrum
- Referred to the game of rugby to stress the importance of teams
- Their research showed that teams requires autonomy to achieve excellence

# Scrum is…

A lightweight framework for project management.

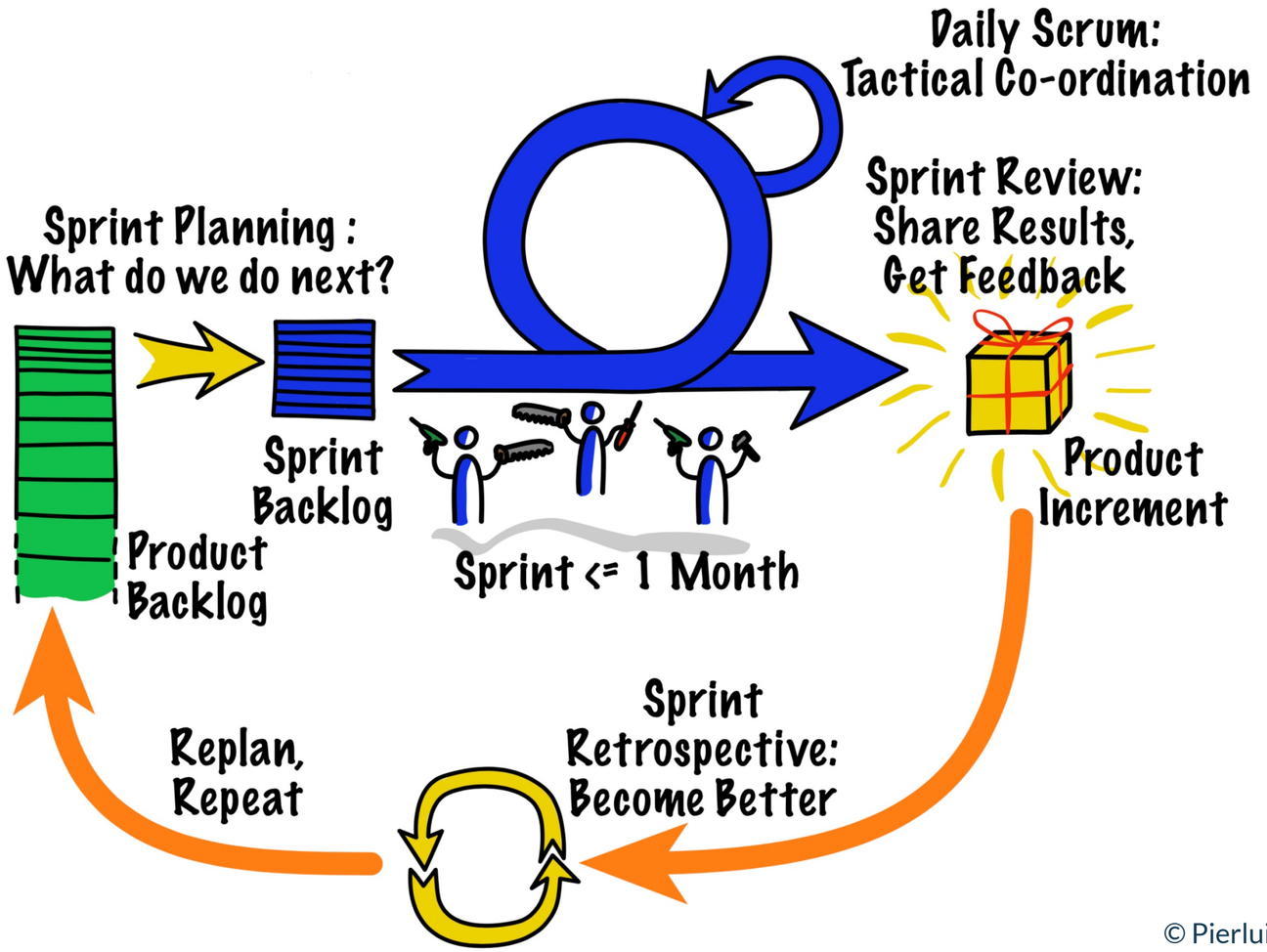
- Few but clearly defined roles
- Self-organizing team
- Time boxed iterations
- Founded on empirical process control theory
- Simple to understand
- Difficult to master



Scrum is not a process or a technique for building products; rather, it is a framework within which you can employ various processes and techniques. Scrum makes clear the relative efficacy of your product management and development practices so that you can improve.

Ken Schwaber and Jeff Sutherland - The Scrum Guide™

# Process Overview



# References

**Agile Manifesto**

https://agilemanifesto.org

**Scrum Guide**

Jeff Sutherland, Ken Schwaber
https://scrumguides.org