Agile and Lean Software Development

Thinking and Practices

- Kaustabh Debbarman, Nokia

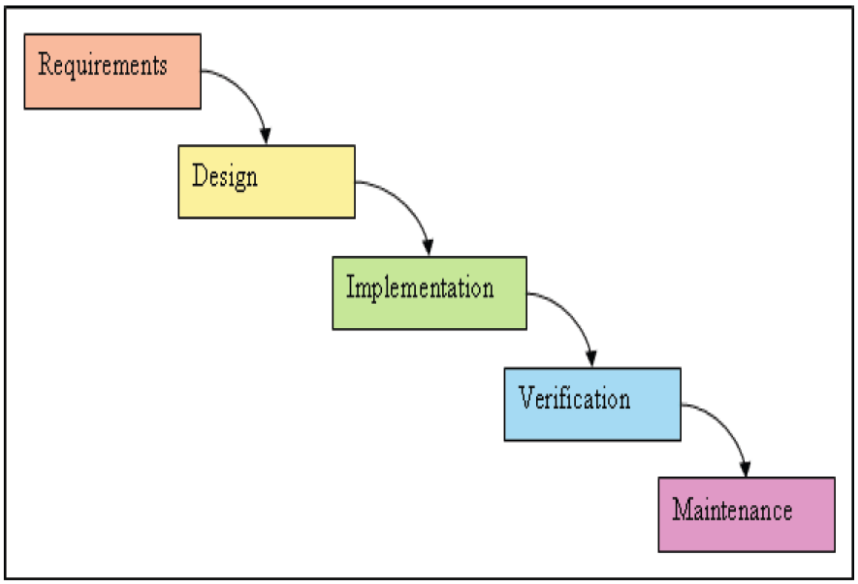
NOKIA

Lets start with a quick poll ☺

How familiar are you with “agile / lean / scrum / XP” ?

2010 Chevrolet Agile

How have we been always doing software development ?



Adoption of Waterfall

Helped drive down the failure rate of software development projects

But

Around 70 percent* of software projects using this methodology fail to meet their objectives

Waterfall software projects have less than half the success rate (66 percent) of going over Niagara Falls in a barrel ☺

* According to http://www.serena.com/docs/repository/solutions/intro-to-agile-devel.pdf

False Assumptions

The harder we plan and analyze in the beginning, the less there's change in the project

It is possible to "collect" or even "know" all the requirements up-front

Product development process can be defined as a predictable and repeatable process

It's possible to transfer information effectively in written documents without much of human contact.

The reality

There is change always. Uncertainty is best reduced learning iteratively both the product and the process

Requirements evolve as our knowledge increases – based on experience & feedback

New product development is an evolving and adaptive process

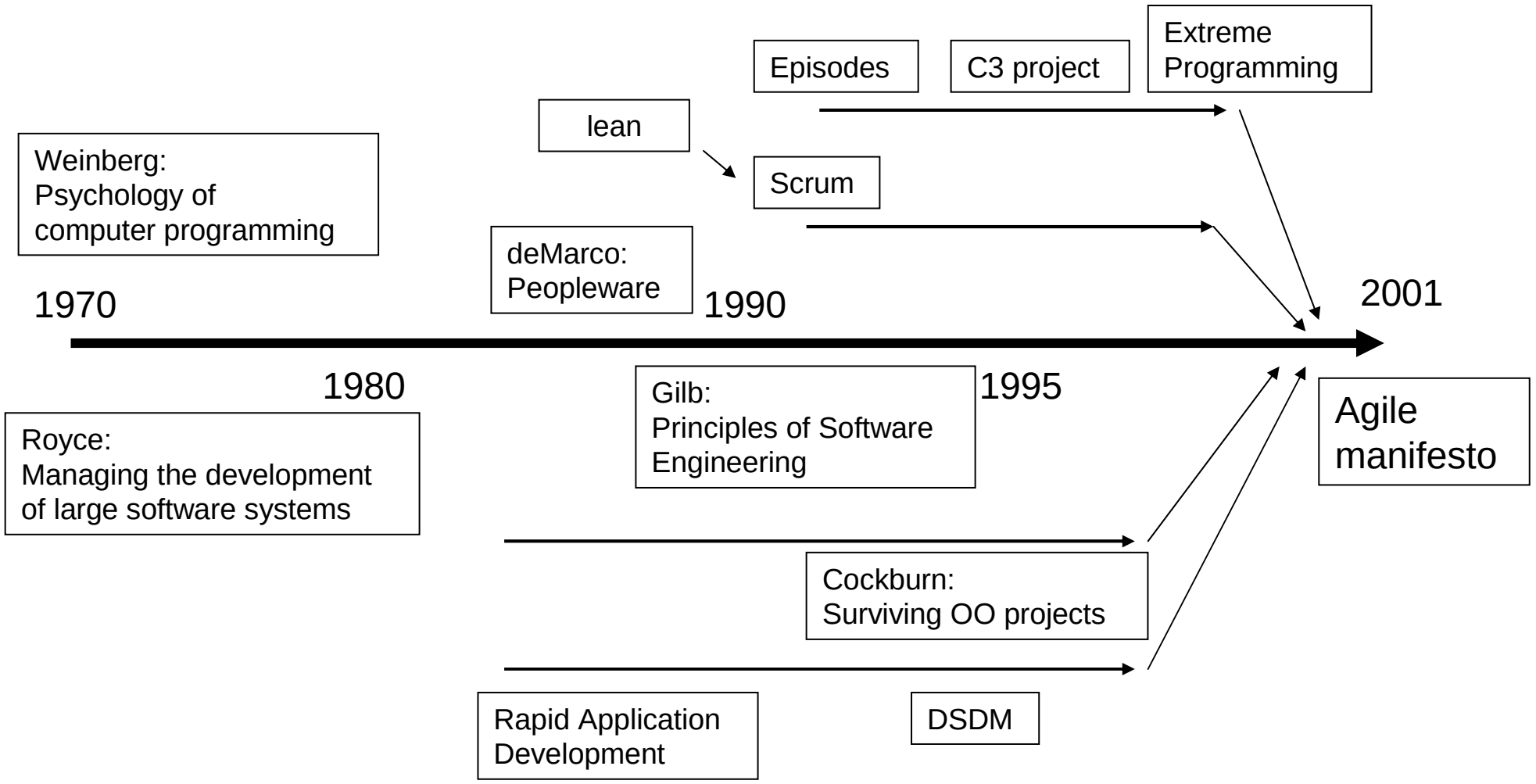
Essential knowledge is lost in every handover and human interaction is needed to overcome it.

And then came the Agile Manifesto

- A set of values published as a manifesto in 2001 with an initial 17 signatories.
- It was a significant departure from the heavyweight document-driven software development methodologies (such as waterfall) in general use at the time.
- The publication of the “Manifesto for Agile Software Development” didn’t start the move to agile methods.
    - Rather it signaled the industry acceptance of the values

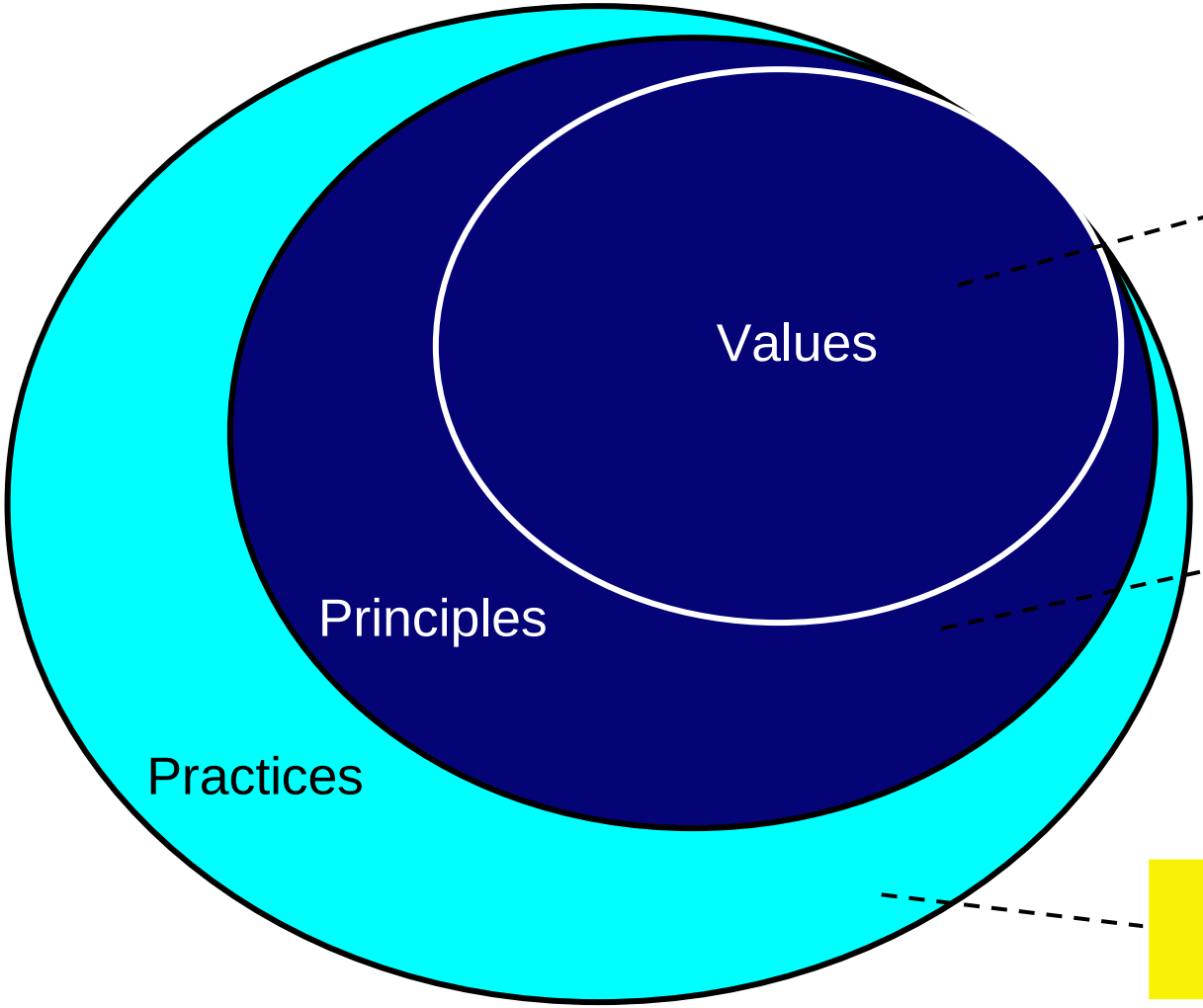
Agile Background

The publication of the "Manifesto for Agile Software Development" didn't start the move to agile methods. Rather it signaled the industry acceptance of the values



Source : Unknown

Agile thinking explained



Defines the core set of most important beliefs of what is truly important

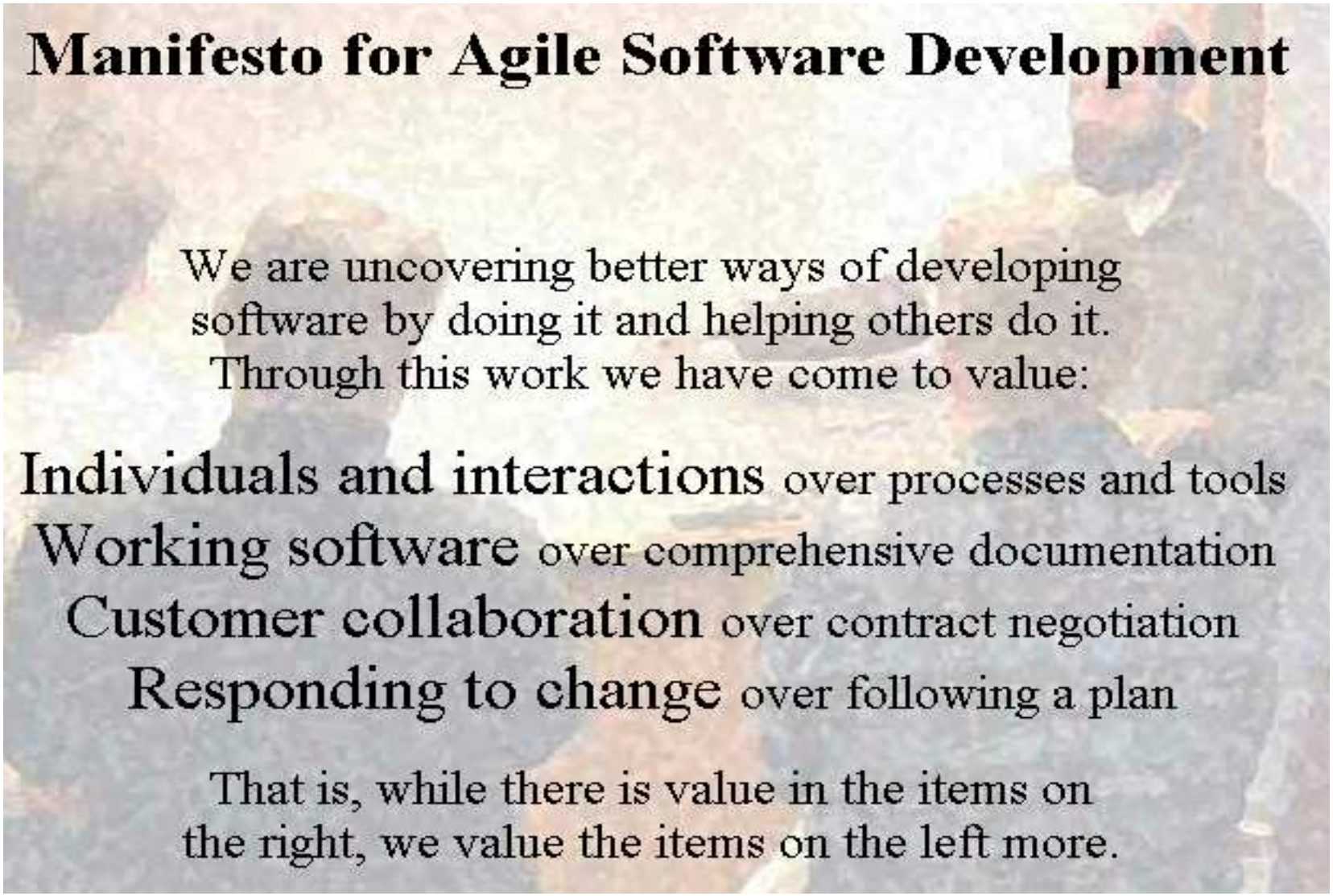
Defines a set of ways to meet the values

Defines in detail how this is implemented in practice

Values and Principles are commonly agreed; Practices are team or project dependent

Source : Pekka Abrahamsson

Values


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.

Agile Principles 1-6

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 a shorter timescale.
4. Business people and developers must **work together** daily throughout the project
5. Build project 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 development team is **face-to-face conversation**.

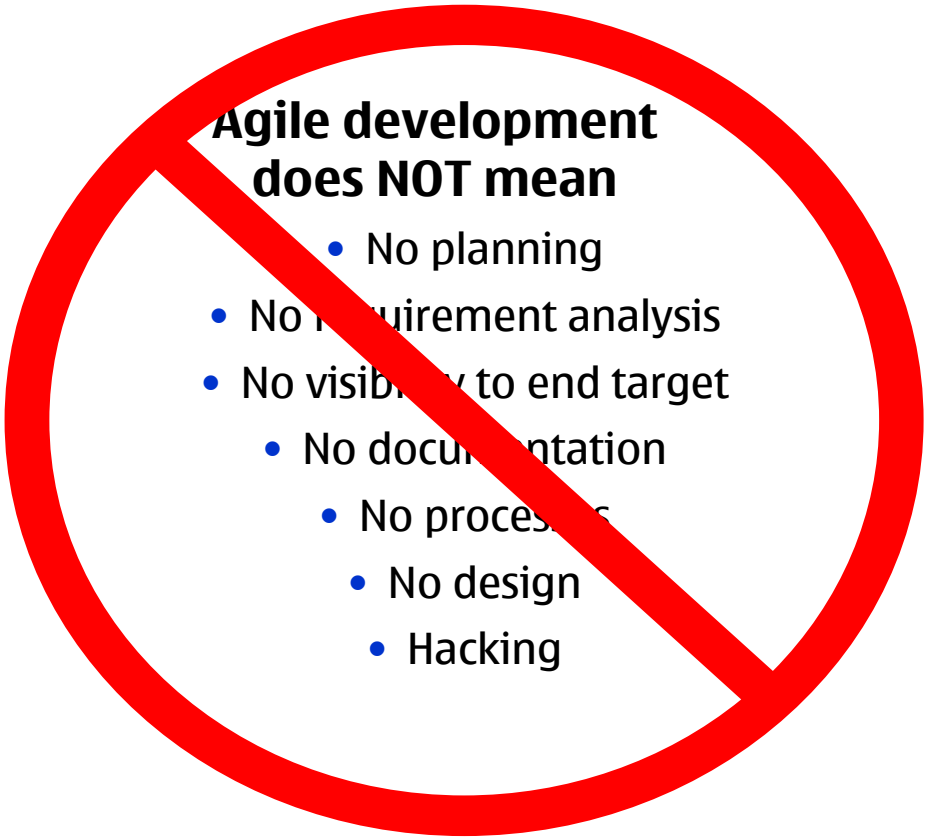
Agile Principles 7-12

7. **Working software** is the primary measure for 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 reflect on how to become more effective**, then tunes and adjusts its behavior accordingly.

You can't do Agile, You are Agile

- Agile is about Values and Principles
  - **People and organizations can not do Agile – they can be Agile**
- Working practices can be Agile and fit into Values and Principles
  - **Teams can do Scrum or XP or whatever**

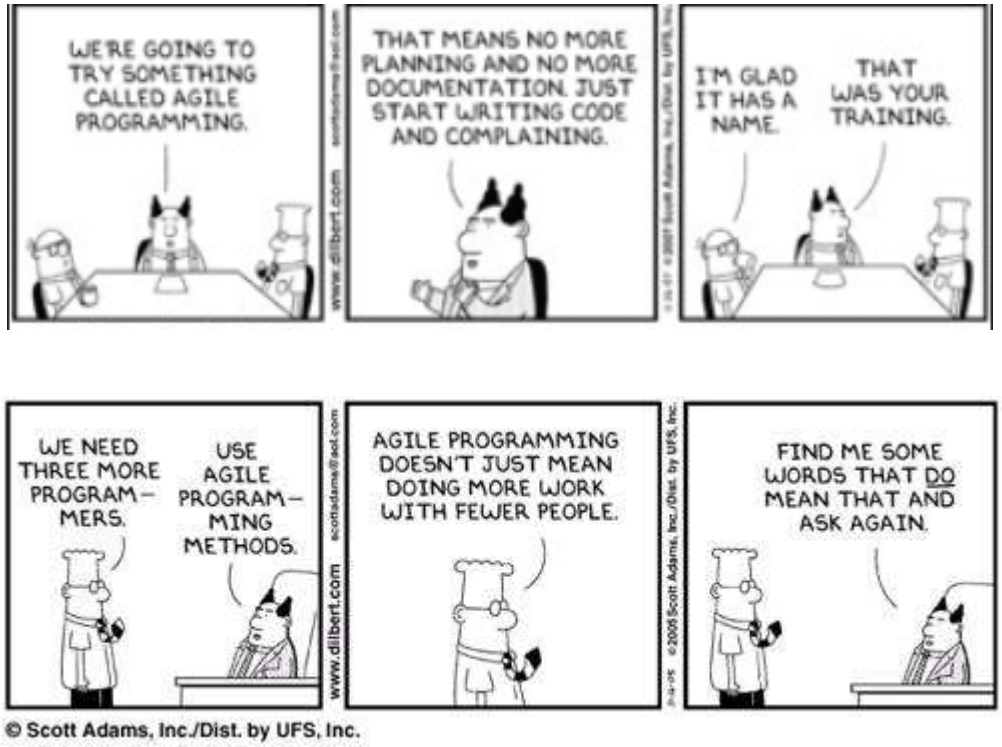
Agile is NOT Ad hoc



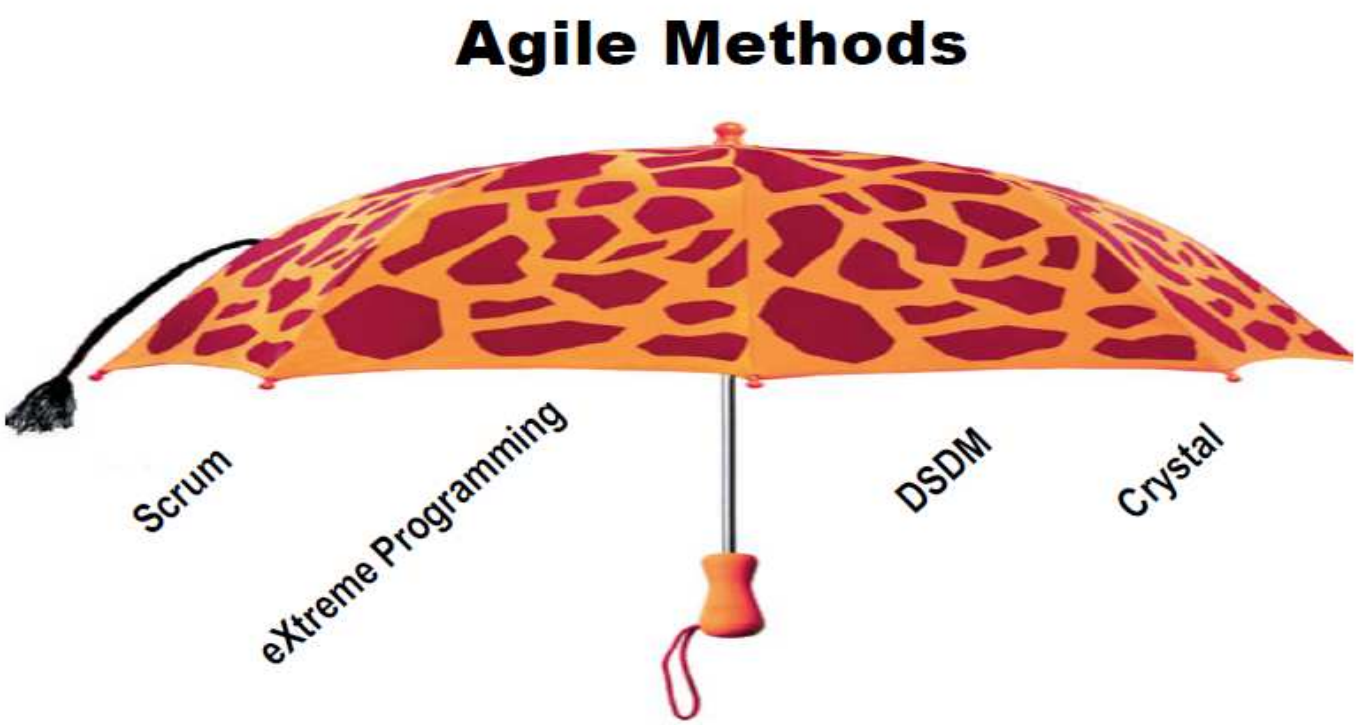
- Agile development means
  - Adaptive planning
  - Evolving requirements
  - Good design done in co-operation
  - Sufficient documentation for the team, customers and maintenance
  - High degree of self-discipline and skillful software engineering
  - Adaptive processes

Disciplined and structured but not a sequential life-cycle

Dilbert & Agile

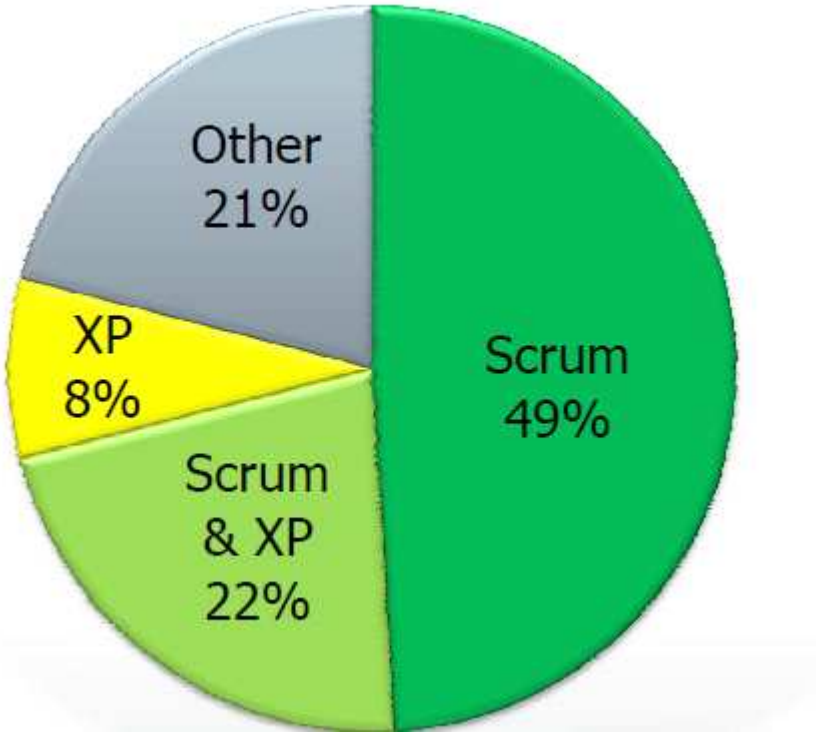


Umbrella of Agile methods



Source : Jeff Sutherland

70% of agile companies use scrum or try to



Source : State of agile Development survey ‘2008

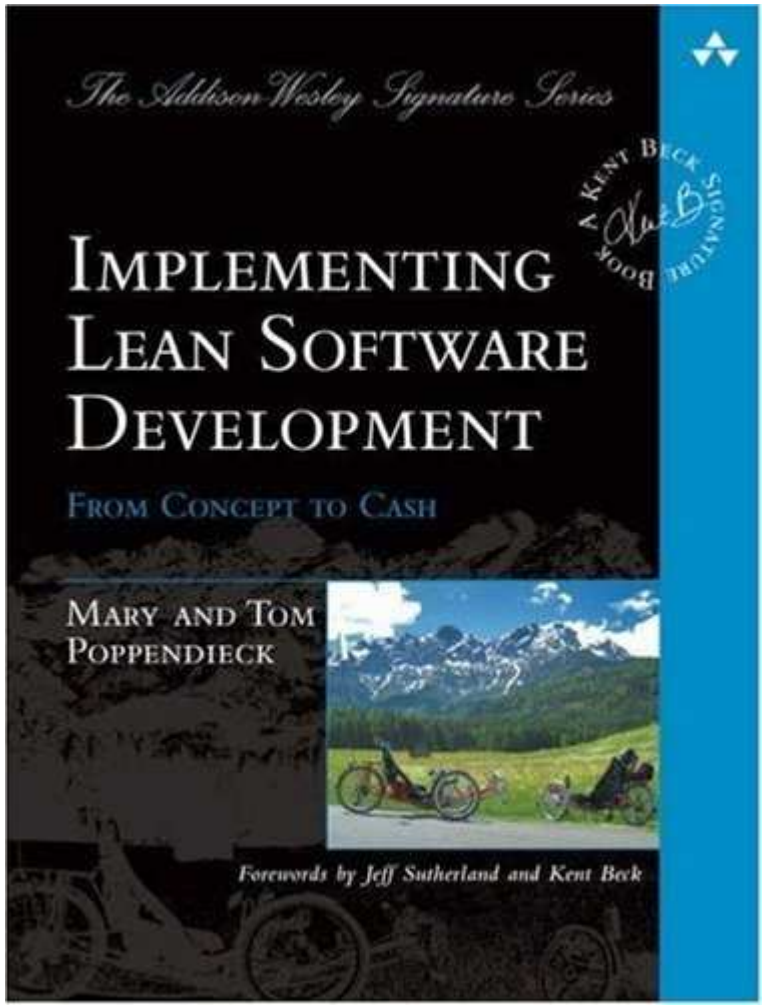
Lean thinking

- *Lean* originated from a benchmarking study published in 1990 between Japanese and US/European car manufacturers
  - "The machine that changed the world"
- The study showed that for certain Japanese manufacturers:
  - Average productivity was about 50% higher
  - Average quality was about 50% higher
  - Product development was about 40% faster with less people
- The study introduce the word "lean" for the different production/thinking style.
  - Lean is heavily based on Toyota Production System (TPS)
- Agile & Lean development has strongly been influenced by lean production.

LEAN IS IN

Principles of Lean SW Development

1. Eliminate Waste
2. Build Quality in
3. Create Knowledge
4. Defer Commitment
5. Deliver Fast
6. Respect People
7. Optimize the whole



Eliminate Waste

Seven Wastes of Manufacturing	Seven Wastes of Software Development
Inventory	Partially Done Work
Extra Processing	Paperwork
Overproduction	Extra Features
Transportation	Task Switching
Waiting	Waiting
Motion	Motion
Defects	Defects

Recognizing Waste

- Eliminating waste means first we have to recognize waste
  - Waste = something that doesn't add value. We need to develop a keen sense of what value is !
  - Value Stream Mapping

Build Quality in

- Build Quality is not same as "***Test quality in***"
- There are two kind of inspection
  - Inspection to find defects – WASTE
  - Inspection to prevent defects – Essential
- The role of Quality Assurance
  - Not to kill mosquitoes
  - But instead put up screens
- The point is not to eliminate defects, it is to not have them appear in the first place! Defects are waste.
  - If you routinely find defects during verification → process is defective.

Create Knowledge

- Software development is a knowledge creation process.
- Assumption in waterfall :
  - Knowledge exists in the form of requirements and detailed design specifications prior to and separate from coding.
- In practice, a development process focused on creating knowledge will expect the design to evolve during coding and will not waste time locking it prematurely.

Defer commitment

- *A military officer who was about to retire once said: 'The most important thing I did in my career was to teach young leaders that whenever they saw a threat, their first job was to determine the time-box for their response. Their second job was to hold off making a decision until the end of the time-box, so that they could make it based on the best possible data.'*
- In Lean Software Development, decisions are not avoided;
  - they are scheduled and made at the last responsible moment.
  - This assures that all decisions are made in a timely manner, yet they are made with as much information as possible to help make the best decision possible.

Deliver fast: Pull and Flow

- Remove waste by focusing on pull
  - Produce something (both goods and knowledge) just-in-time, when needed, that is, use pull scheduling
- Produce small units and focus on flow
  - Even out the arrival of work
  - Minimize the number of things in process
  - Minimize the size of things in process
  - Establish a regular cadence and continuous flow
- Limit work to capacity
- Optimize throughput – not capacity
  - Stop trying to maximize "resource" utilization

Respect people

- Move the responsibility of decision making to the lowest possible level
- Move from
  - Directed (command and control) → Self-directing and self-managing
- Teams are given general plans and reasonable goals and are trusted to self-organize to meet the goals.

Optimize the whole

- A lean organization optimizes the whole value stream.
  - Sub optimization is bad , but in reality software development is legendary for its tendency to sub optimize.

The Toyota Production System

Taiichi Ohno

(1912-1990)

- Approach to Production
  - Build only what is needed
  - Stop if something goes wrong
  - Eliminate anything which does not add value
- Philosophy of Work
  - Entrust workers with responsibility & authority
  - Go and see for yourself to thoroughly understand the situation.
  - Become a learning organization through relentless reflection and continuous improvement.

"All we are doing is looking at the time line from the moment the customer Gives us an order to the point when we collect the cash. And we are reducing that Time line by removing the non-value added wastes"

Agile or Lean? ☺

- Many commonalities between lean and agile, including:
  - People centric approach
  - Empowered teams
  - Adaptive planning
  - Continuous improvement
- You can't really talk about them being alternatives
  - You are Agile when you do Lean software development.
  - You are Lean when you do Agile software development.

Scrum?

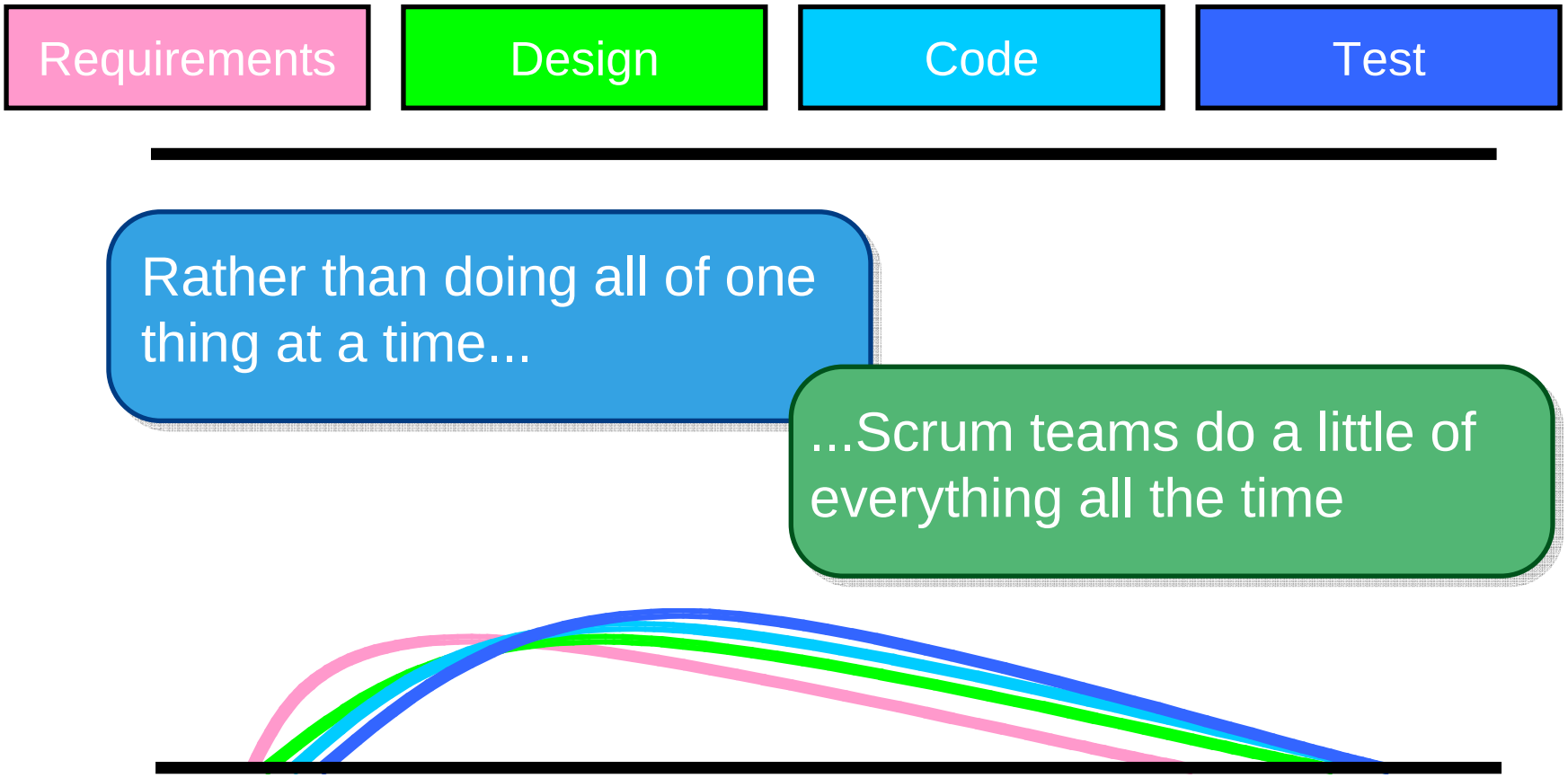
We're losing the relay race

"The… 'relay race' approach to product development…may conflict with the goals of maximum speed and flexibility. Instead a holistic or 'rugby' approach—where a team tries to go the distance as a unit, passing the ball back and forth—may better serve today's competitive requirements."

Hirotaka Takeuchi and Ikujiro Nonaka, "The New New Product Development Game", *Harvard Business Review*, January 1986.

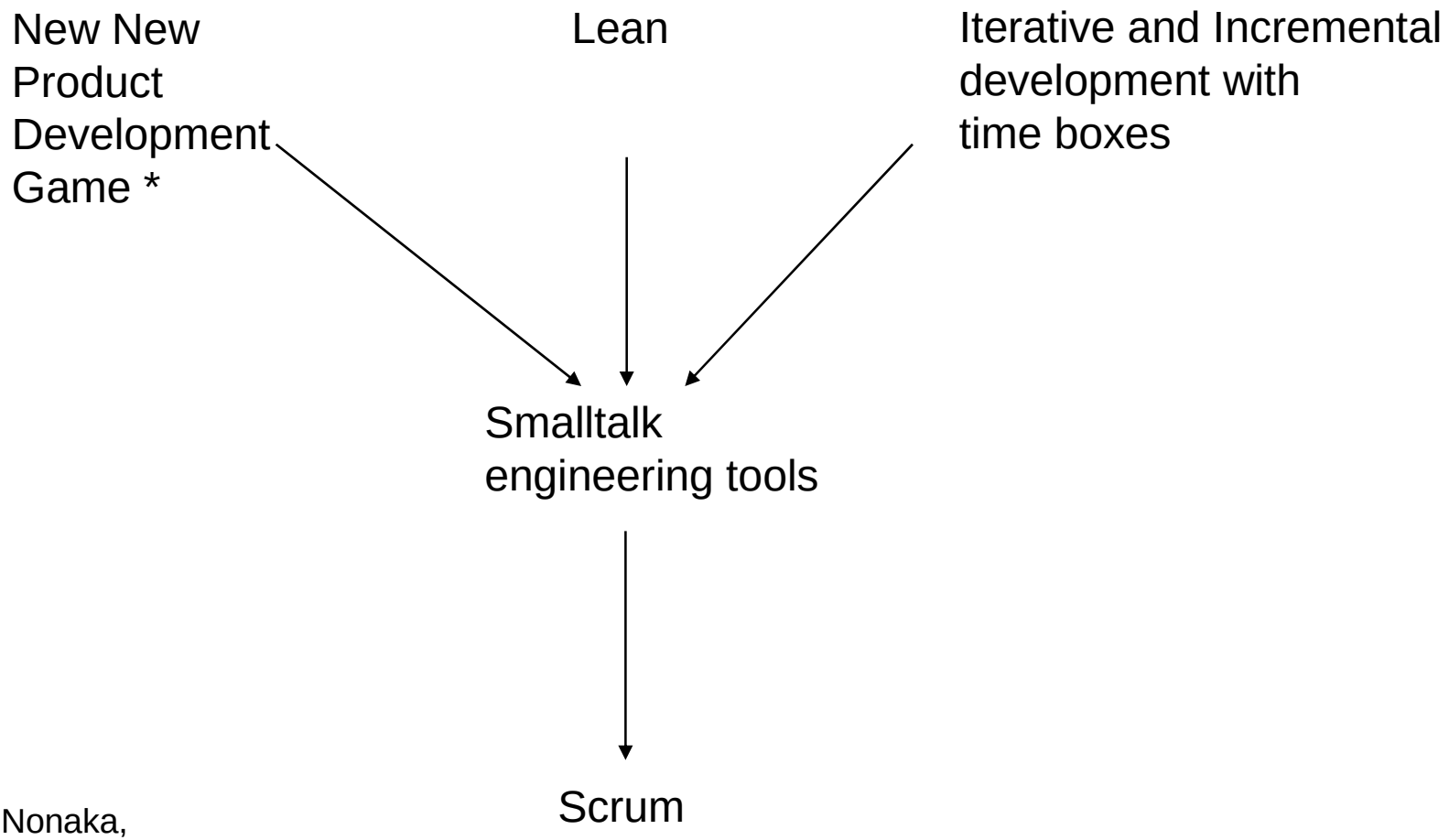
NOKIA

Sequential vs. overlapping development



Source: “The New New Product Development Game” by Takeuchi and Nonaka. *Harvard Business Review,* January 1986.

Concepts that influenced Scrum



*Takeuchi and Nonaka,
Harward Business Review Jan 1986

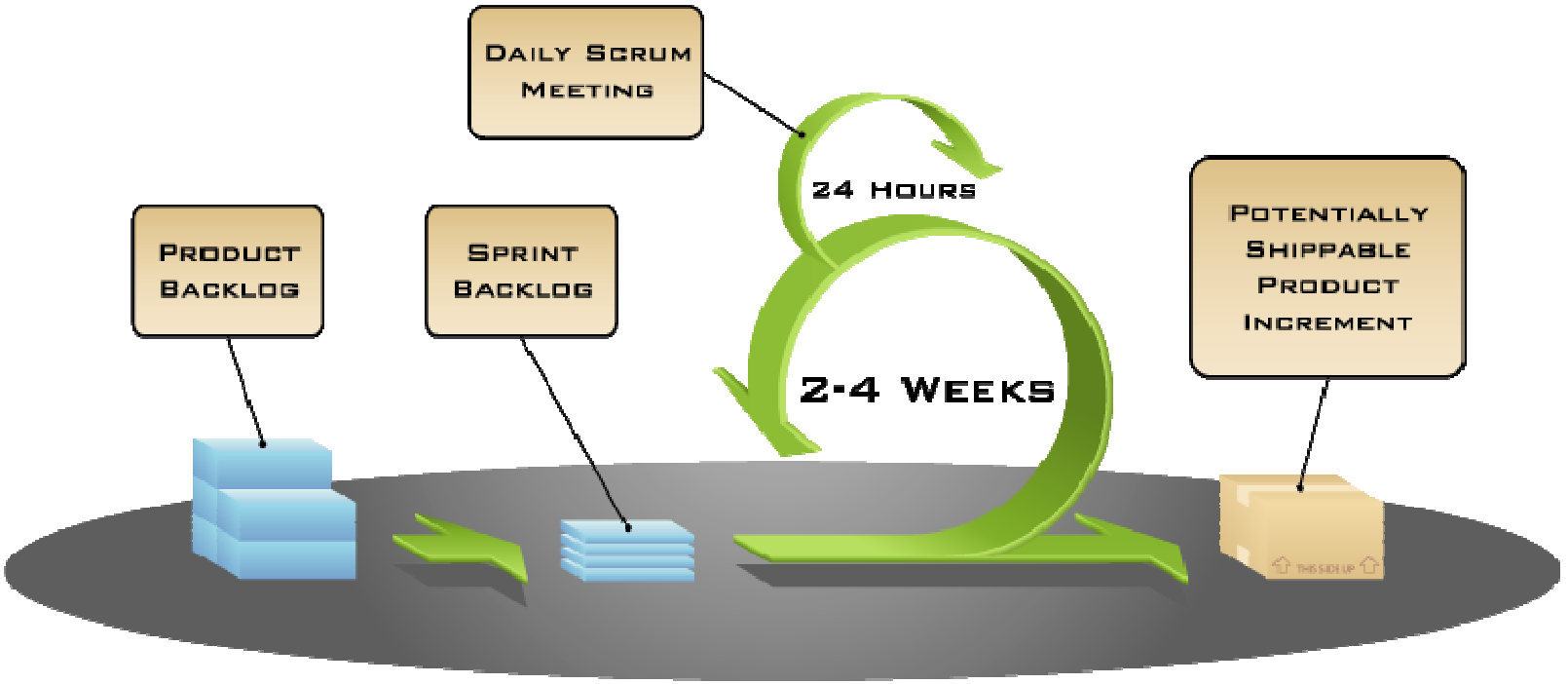
What is Scrum?

"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. The role of Scrum is to surface the relative efficacy* of your development practices and highlight the deficiencies and impediments in the organization so that you can improve upon them **while providing a framework within which complex products can be developed**"

- *Ken Schwaber, Scrum Guide (May 2009)*

**efficacy : the power to produce an effect*

The “simple” Scrum picture



All is there to Scrum

- 3 Artifacts
  - Product Backlog
  - Sprint Backlog
  - Burn-down charts
- 3 Roles
  - Product Owner
  - Scrum-master
  - The team
- 3 Meetings(time boxed)
  - Sprint Planning meeting
  - Daily Scrum
  - Sprint review (demo + retrospective)

- Scrum projects make progress in a series of “sprints” which are time-boxed.
- Every sprint leads to a *potentially shippable* increment of the product
  - Product is designed, coded, and tested during the sprint

12 steps to Scrum

1. Agree on a PO, SM, and full Team. And on a Product goal.
2. Set a date now for the Sprint Review in 2 weeks and send out invites.
3. Review/define a ranked Product Backlog of features
4. Estimate the Product Backlog items
5. Conduct Sprint Planning with Team and Stakeholders. Complete Sprint Backlog
6. Commit as a team to the Sprint
7. Track status and obstacles daily via the Daily Scrum
8. Track progress using the Sprint Burndown
9. Conduct a Sprint Review; demo done items
10. Conduct a team Retrospective
11. Take action on top impediment
12. GOTO 2

Source: Hubert Smits & Jean Tabaka

Key Ideas in Scrum

- Inspect & Adapt
- Self-organization
- Empowerment
- Definition of Done

Facts about Scrum

- Scrum is something almost impossibly simple that frequently gets overcomplicated.
- Scrum is about common sense
- Scrum has no no redundant or optional parts
- Scrum is extremely hard to do
  - It disrupts the organization to be the best it can be, and it puts facts you'd prefer not to know right in your face.
  - Everything that was **hard** in waterfall engineering practices now has to be done every sprint, and this is incredibly **hard**. It is not impossible, but has to be worked toward over time.

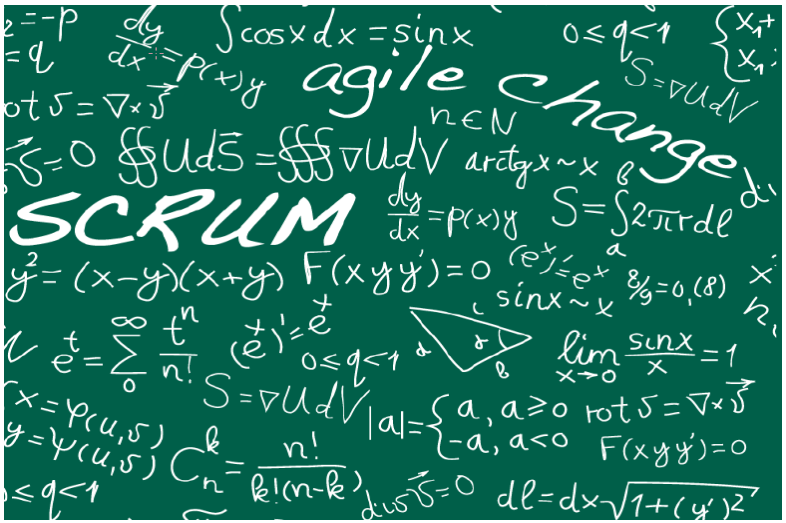
Scrum but

“I’m all for improvement; I just don’t want to change things.”

Confused about where to go?

How about this?

- *Agile / Scrum / Lean for dummies*
  - *Let's talk to each other*
  - *Let's just build it and show you*
  - *Let's trust each other*
  - *Let's respond to what is happening and what we learn.*


agile change
SCRUM


THANK
YOU