2014

Agile/Scrum Introduction

Carlo Criniti

Presenter // Education

- Yverdon-les-Bains
  - Bachelor of Sciences in Telecommunications

heig-vd
Haute Ecole d'Ingénierie et de Gestion
du Canton de Vaud

- Lausanne
  - Master of Sciences in Engineering (TIC)

MSE | MASTER OF SCIENCE
IN ENGINEERING

- Bern
  - Master thesis

swisscom

Presenter // Experience

- Lausanne
  - Solution Engineer
  - Part-time Agile “consulting” (training, coaching, Agile SIG, Poppendieck’s)
- Zurich
  - Project Manager
  - Scrum Master, Agile trainer/coach, Solution Devliery Framework eng.
- New York
  - Agile Trainer & Coach
  - Agile Training delivery (Scrum, Kanban, Scaled Agile), training creation, Agile coaching

CREDIT SUISSE

CREDIT SUISSE

CREDIT SUISSE

Presenter // Currently

- Zurich (→ London)
  - Agile Trainer & Coach
  - Agile Training delivery (Scrum, Kanban, Scaled Agile), training creation, Agile coaching
  - Program transformation

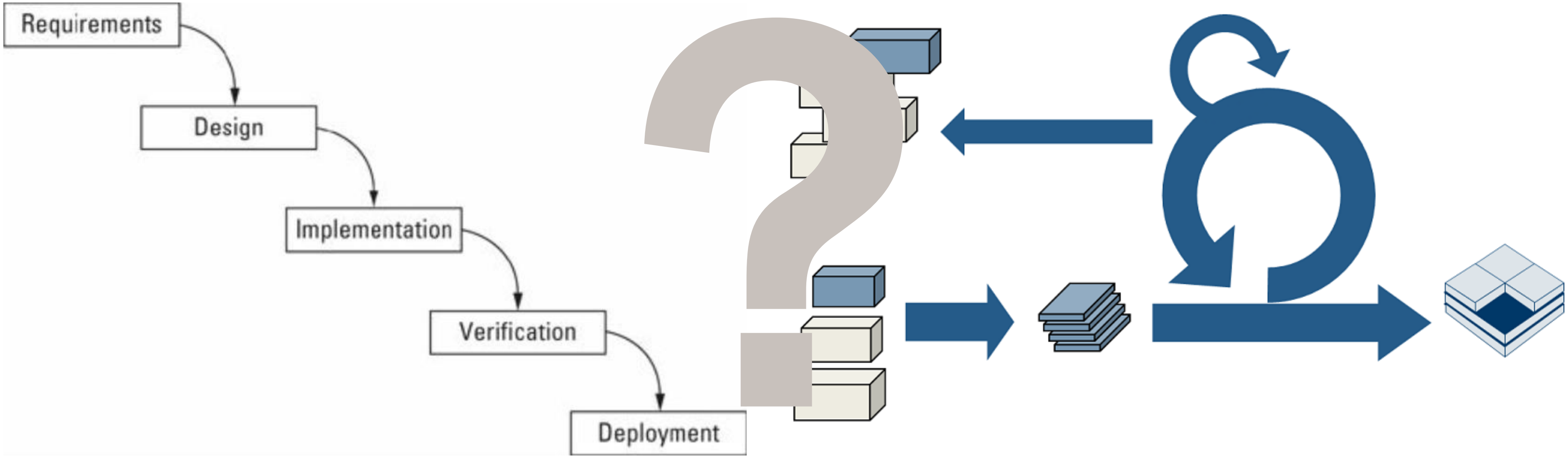
CREDIT SUISSE

Ted and his new project

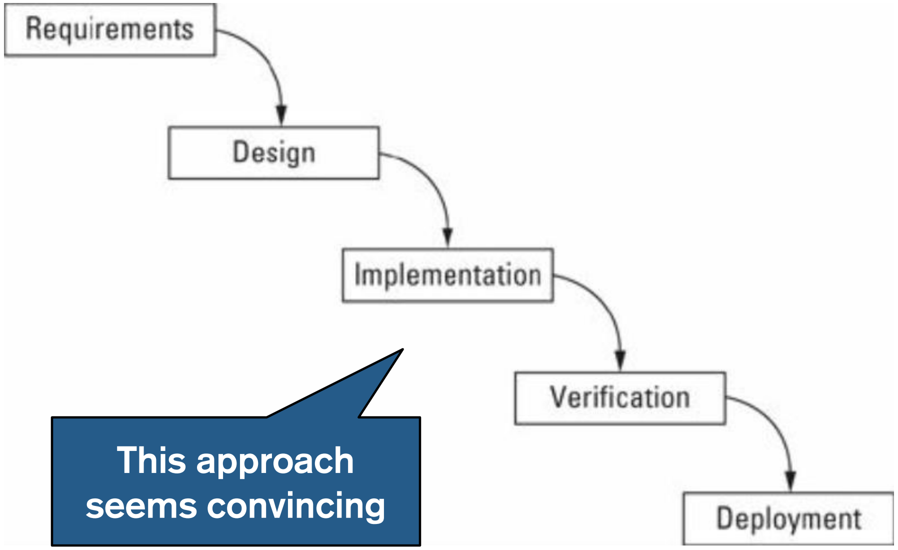
Context project // Ted

- Ted is an IT Project Manager in FakeCorp
- Build a solution to manage travel expenses
- Allocated budget: 4 millions USD
  - 2 years
  - 7 FTE

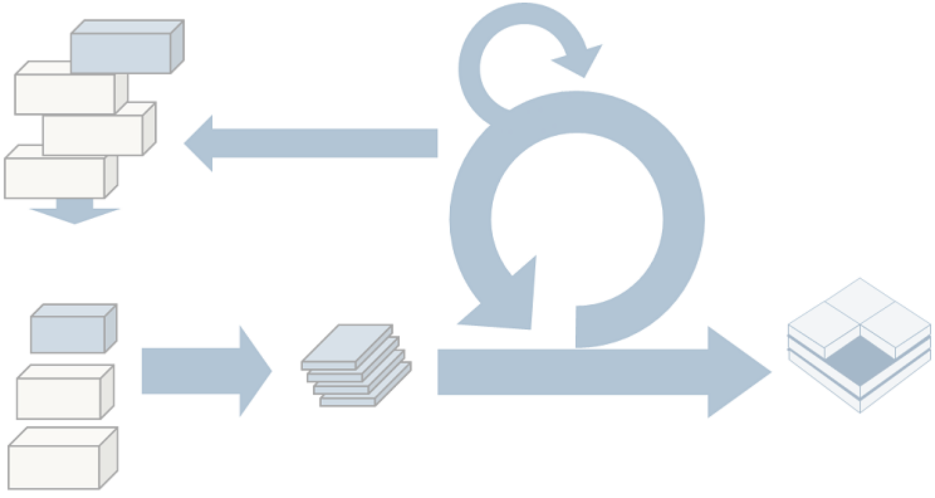
Context project // Ted has to choose



Context project // Ted has to choose

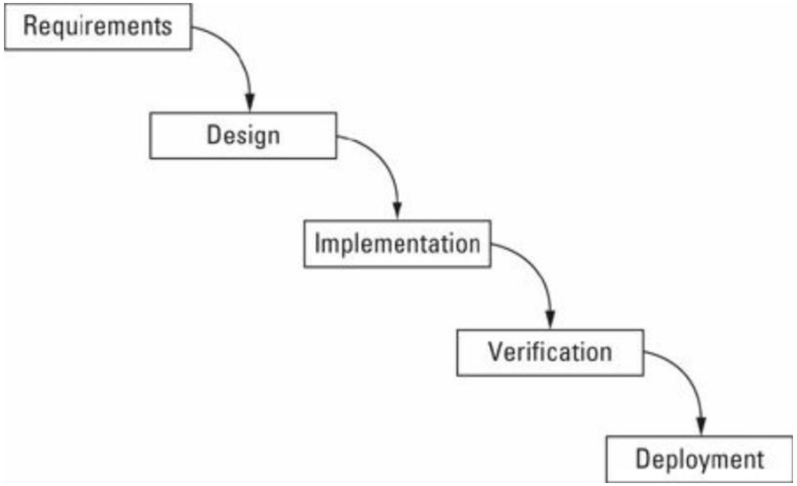


This approach seems convincing



Context project // Ted reassures himself

- Seems to be a great idea to
  - Plan the solution at the beginning
  - Be sure about the requirements before designing the solution
  - Designing the entire solution before building it
  - Test the entire solution, when it is integrated
  - Deliver it to the customer
  - Receive congratulations

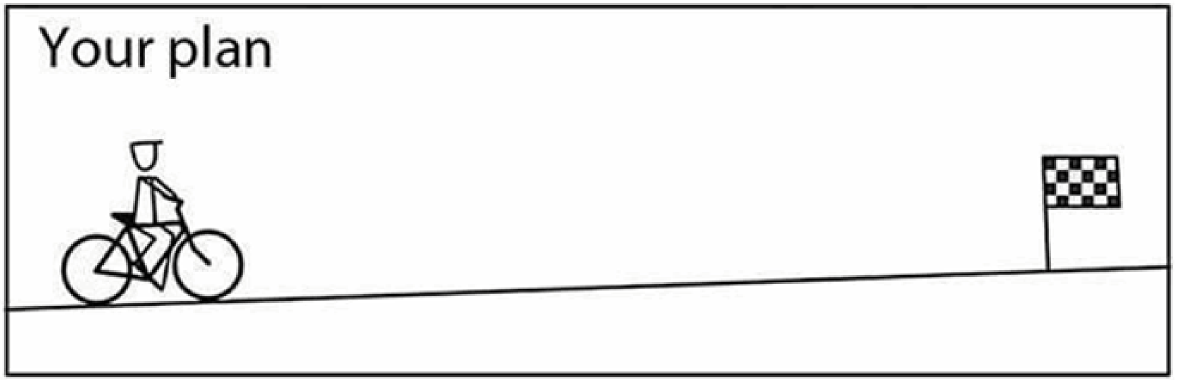


Context project // Ted plans

- Let's plan according to the book

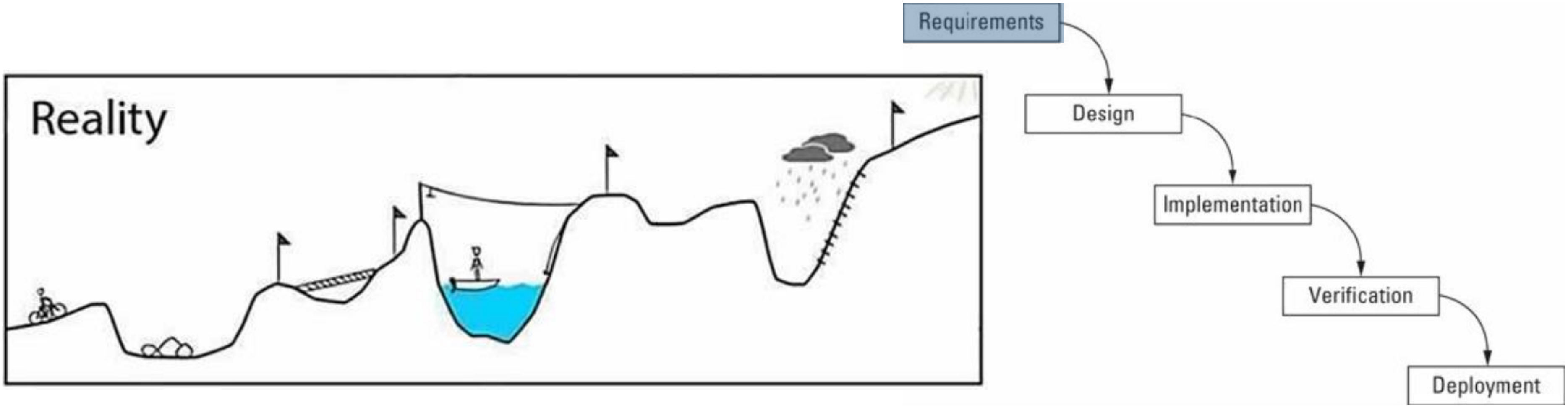
	24 months				
Phase	Analysis	Design	Code	Verification	Deployment
Percent	10%	25%	40%	20%	5%
Months	2.5	6	9.5	5	1

- And start this project
  - Everything works according to plan



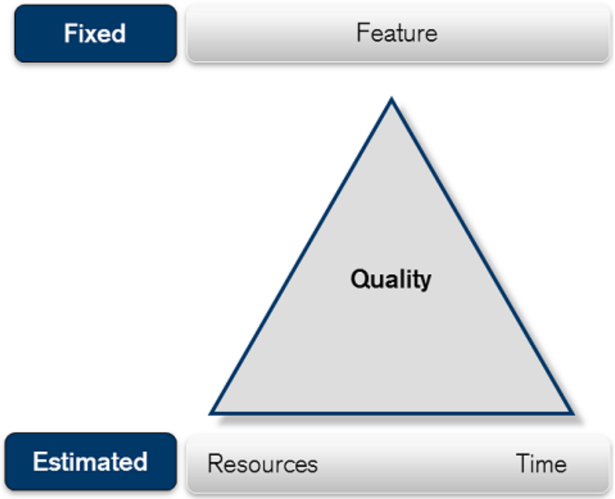
Context project // Early problems

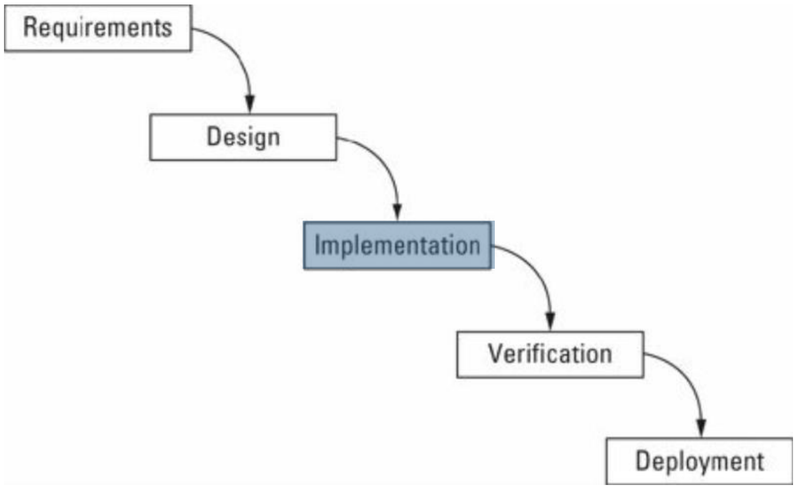
- Requirements take more time to get than planned
  - Lack of commitment
  - Frequent changes



Context project // Implementation is rushed

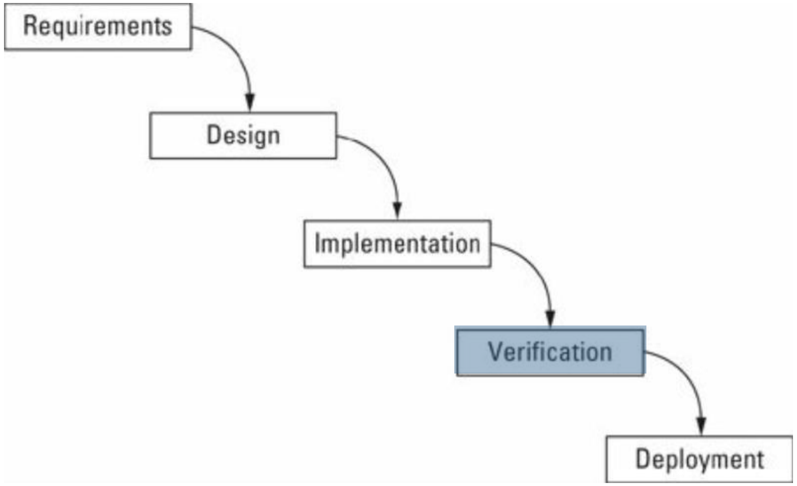
- Implementation starts late
  - Have to implement everything in less time than planned
  - New issues emerge
  - Verification will be reduced in order to finish implementation





Context project // Verification starts late

- Due to late implementation
  - Verification starts very late
  - Verification has to be reduced to finish on time
  - Integration is a nightmare
  - Testing reports many issues
  - Solving issues creates new ones
  - Team is working during weekends

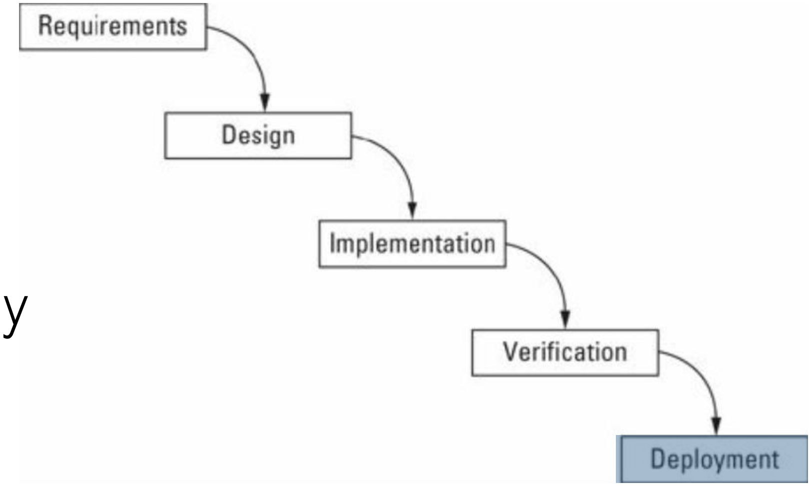


Context project // Deployment

- Bad quality (reduced Verification phase)

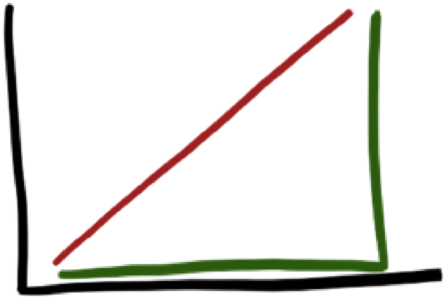
- Late delivery (bad quality and team morale made things even worse)

- Customer is not satisfied
  - Customers needs have changed in 2 years
  - Requirements were not understood properly by developers
  - Needs were not understood properly by BAs
  - Customers did not know what they really wanted before seeing it

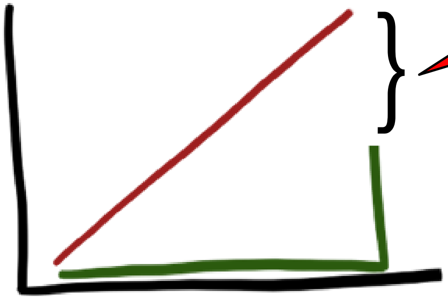


Context project // Results

- Cost vs Value delivered over time
  - Cost
  - Value delivered



Planned

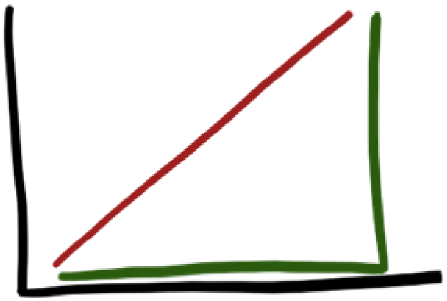


Achieved

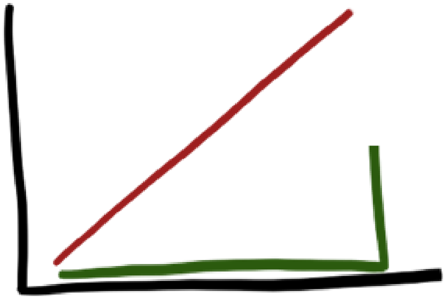
Difference between what the customers thought they wanted to have and what they really wanted

Context project // Results

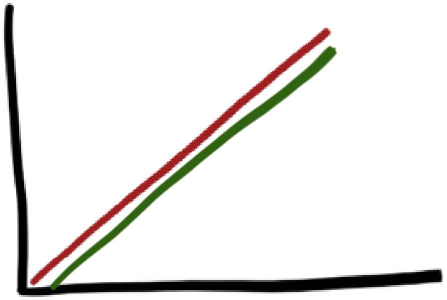
- Cost vs Value delivered over time
  - Cost
  - Value delivered



Planned

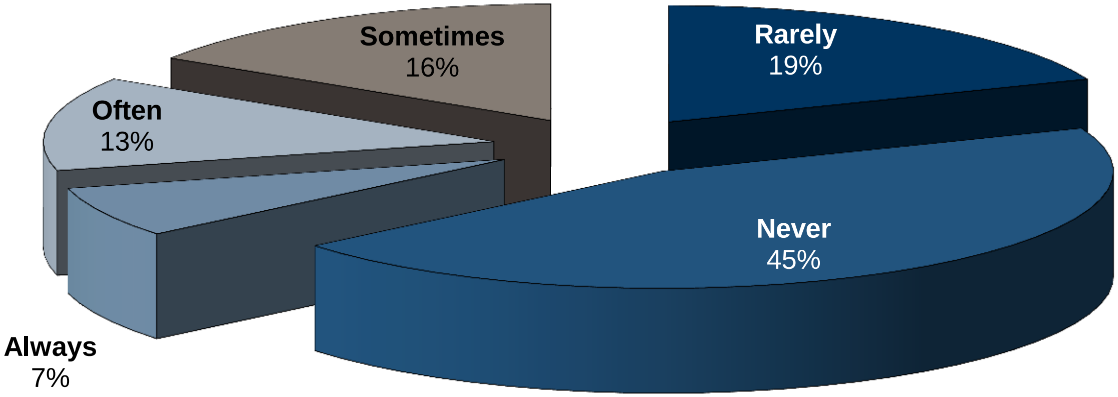


Achieved



Ideal

Context project // Ted’s frustration

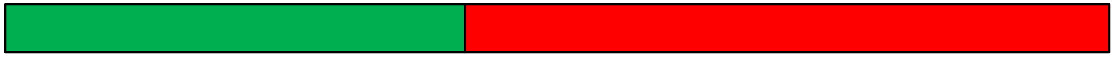


Standish Group Study Reported at XP2002 by Jim Johnson. Chairman

Context project // Reality

- Reality is different
  - Ted saw his budget cut in half

	24 months				
Phase	Analysis	Design	Code	Verification	Deployment
Percent	10%	25%	40%	20%	5%
Months	2.5	6	9.5	5	1

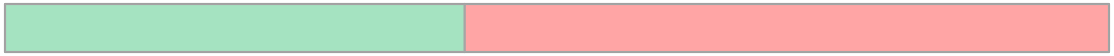


- Code delivered
  - Many features started (interfaces, …)
  - None is finished or usable
  - No value for the customer

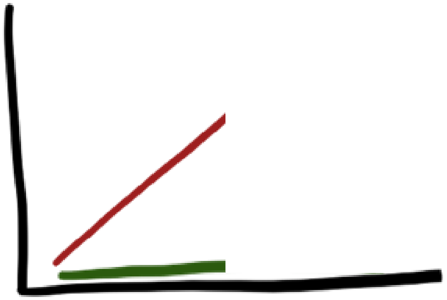
Context project // Reality

- Reality is different
  - Ted saw his budget cut in half

	24 months				
Phase	Analysis	Design	Code	Verification	Deployment
Percent	10%	25%	40%	20%	5%
Months	2.5	6	9.5	5	1



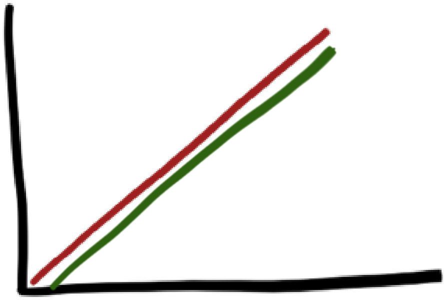
- Code delivered
  - Many features started (interfaces, …)
  - None is finished or usable
  - No value for the customer



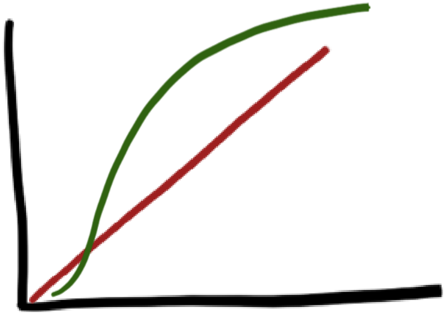
Reality

Agile and Scrum

Agile // Value proposition

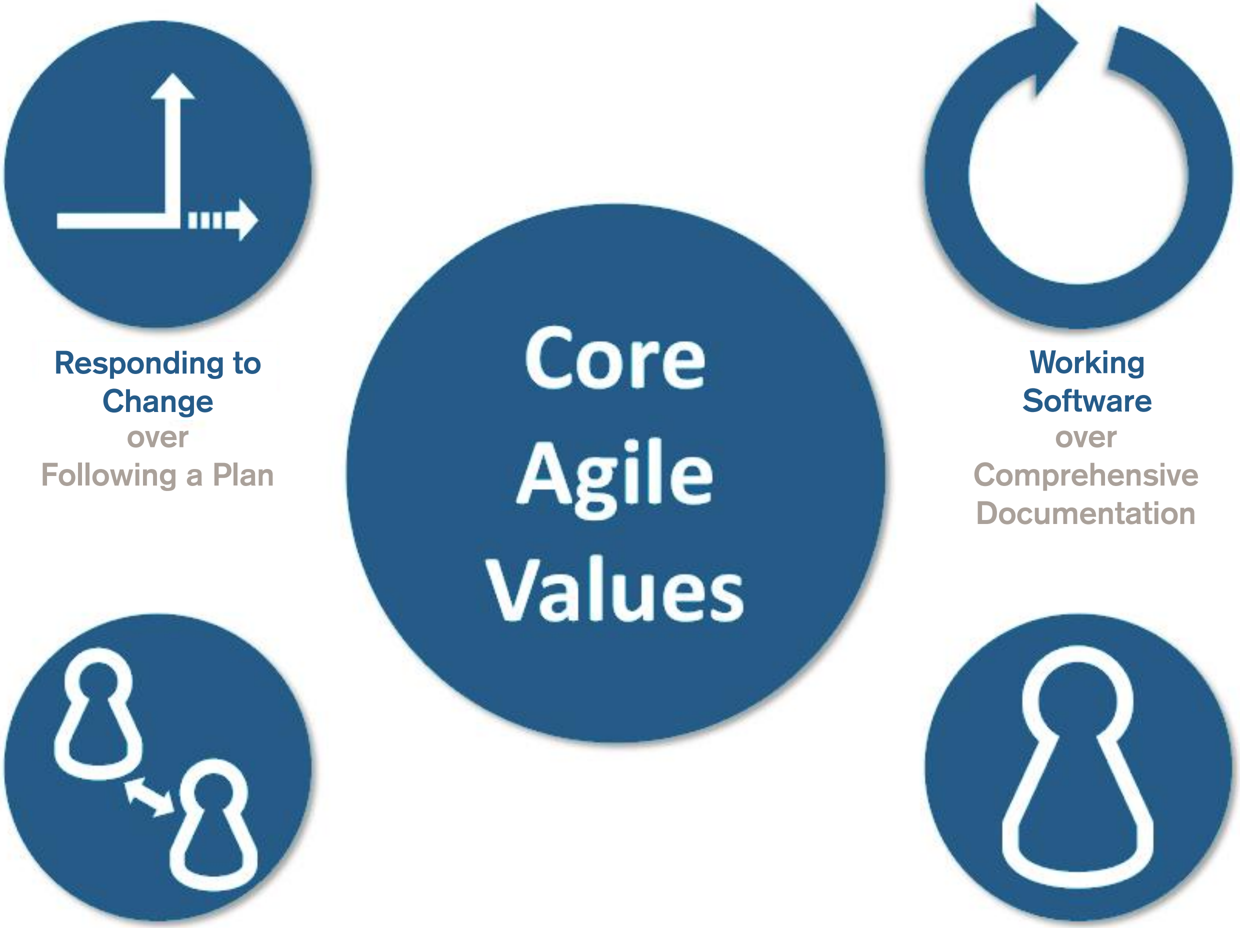


Ideal



Agile

Agile // Based on the Agile Manifesto



Core Agile Values

Responding to Change
over
Following a Plan

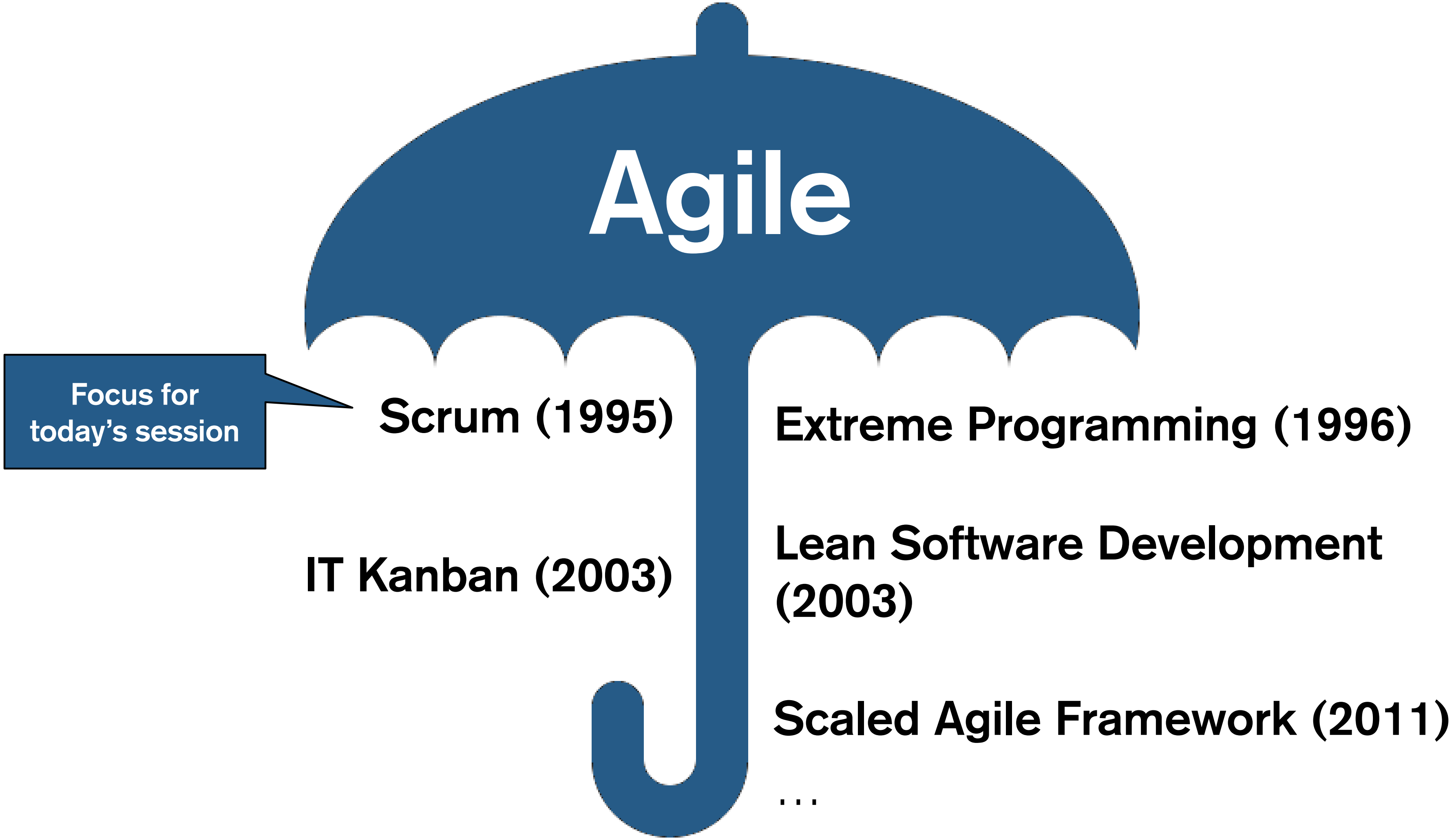
Working Software
over
Comprehensive Documentation

Customer Collaboration
over
Contract Negotiation

Individuals & Interactions
over
Processes and Tools

http://agilemanifesto.org/

Agile // Methodologies



Agile

Focus for today's session

Scrum (1995)

IT Kanban (2003)

Extreme Programming (1996)

Lean Software Development (2003)

Scaled Agile Framework (2011)

…

Agile // Scrum roles

- **Product Owner**

  Owns the product and is the driver for the vision. Selects what has to be built (creates the stories and prioritize them).

- **Scrum Master**

  Coaches and supports the Team and the Product Owner.

- **Team (7±2)**

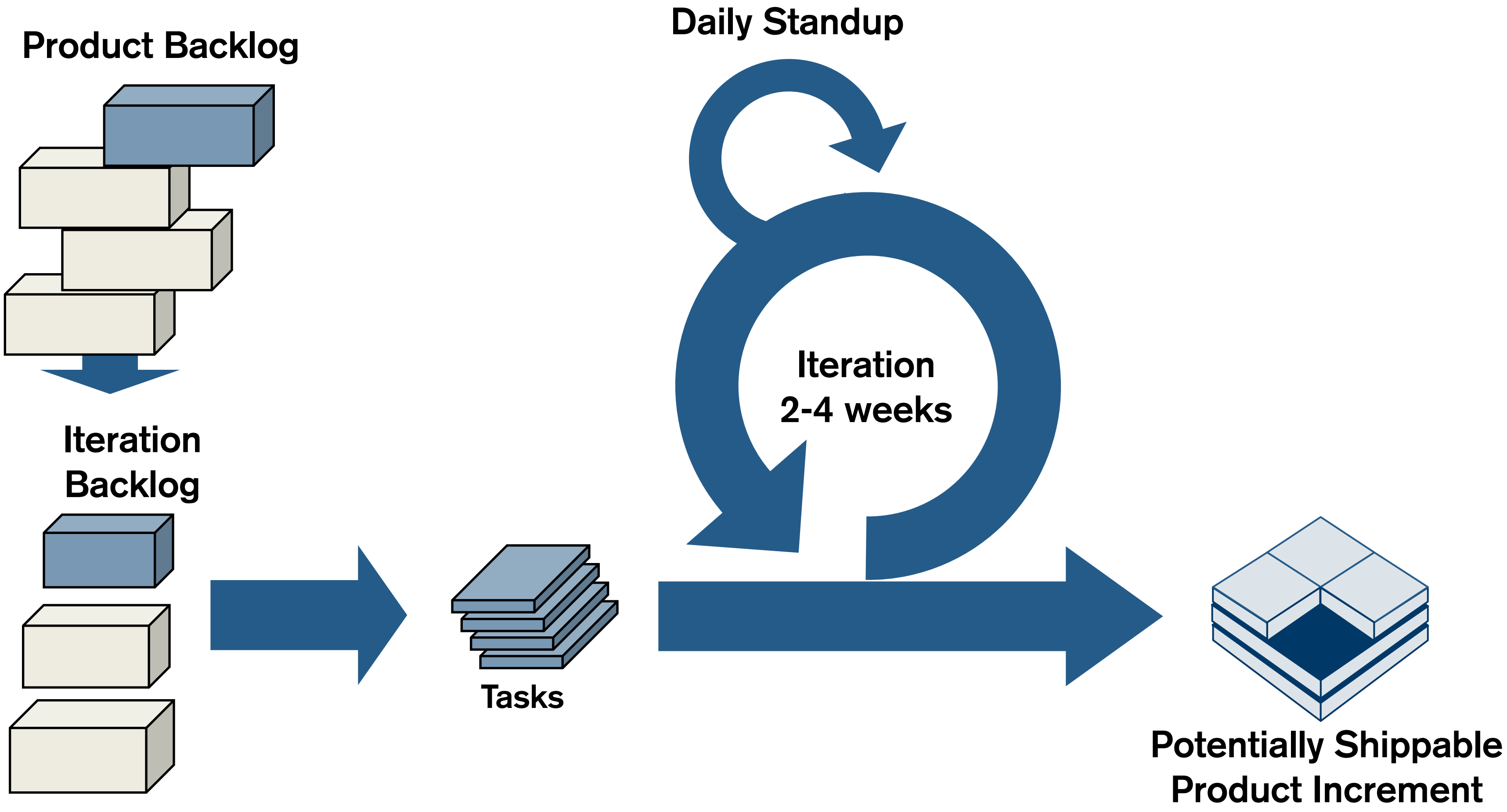
  Self-organizes and is responsible for the results of the iteration. Selects how much and how things will be built (selects how many stories)

- **(Observer and Stakeholders)**

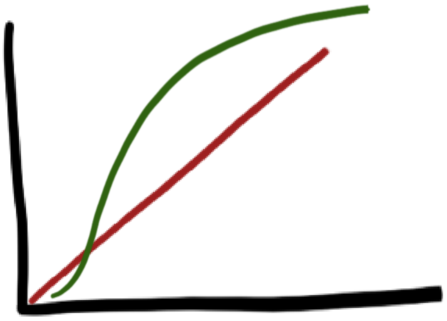
  Support the Product Owner and give the Scrum team the right environment.

Agile // Scrum

Agile // Scrum


Product Backlog
Daily Standup
Iteration
2-4 weeks
Iteration
Backlog
Tasks
Potentially Shippable
Product Increment

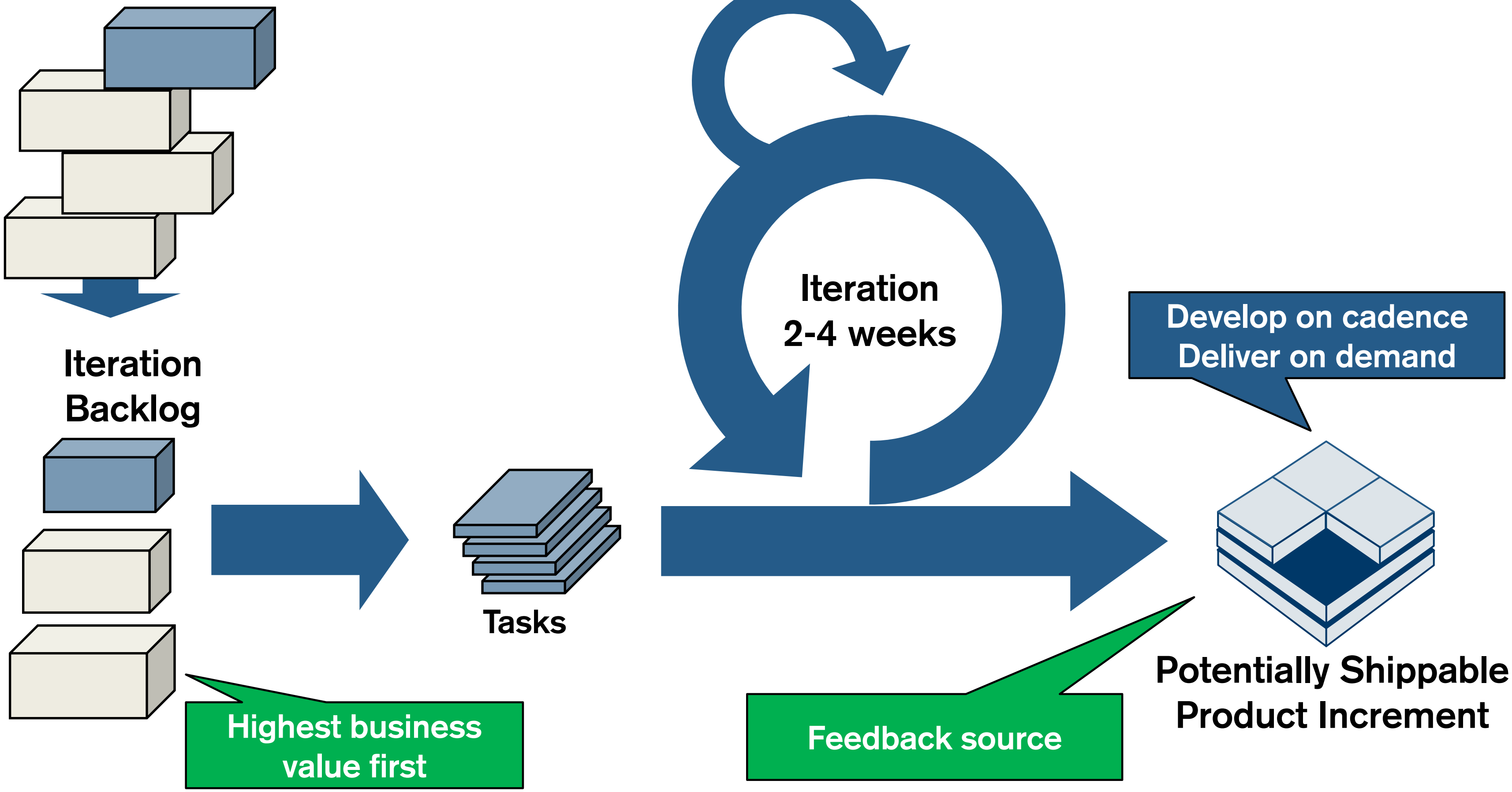
Agile // Scrum



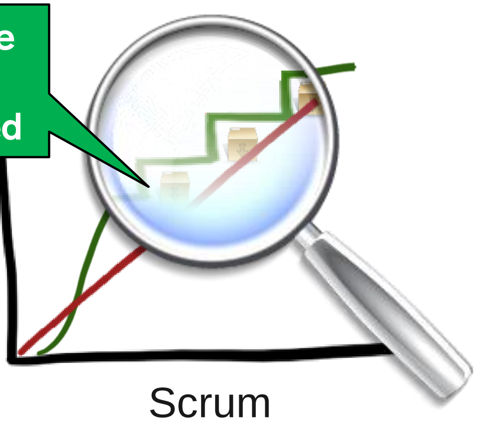
Scrum

Product Backlog

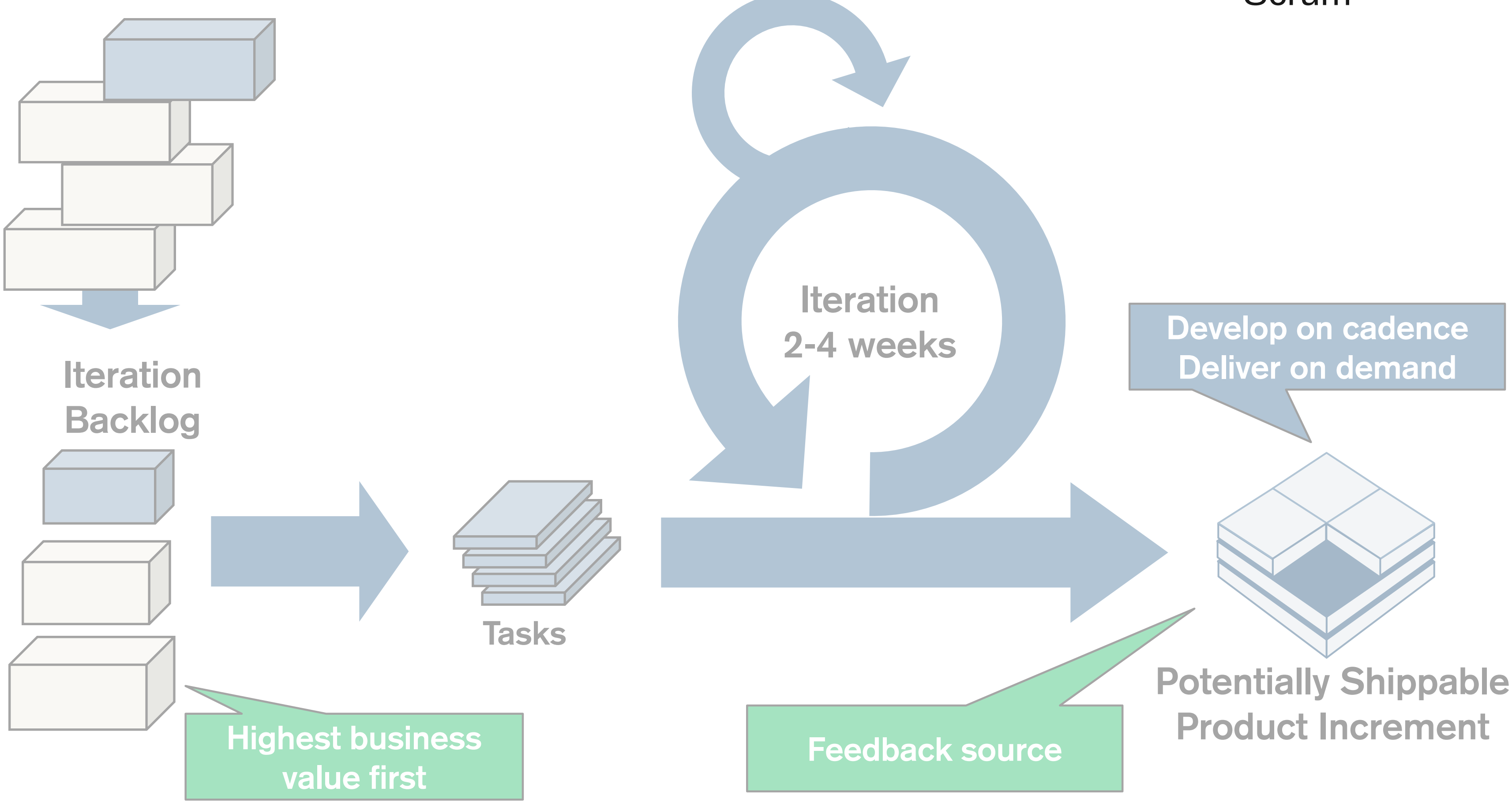
Daily Standup



Agile // Scrum



Incomplete, but usable software, which is tested and documented

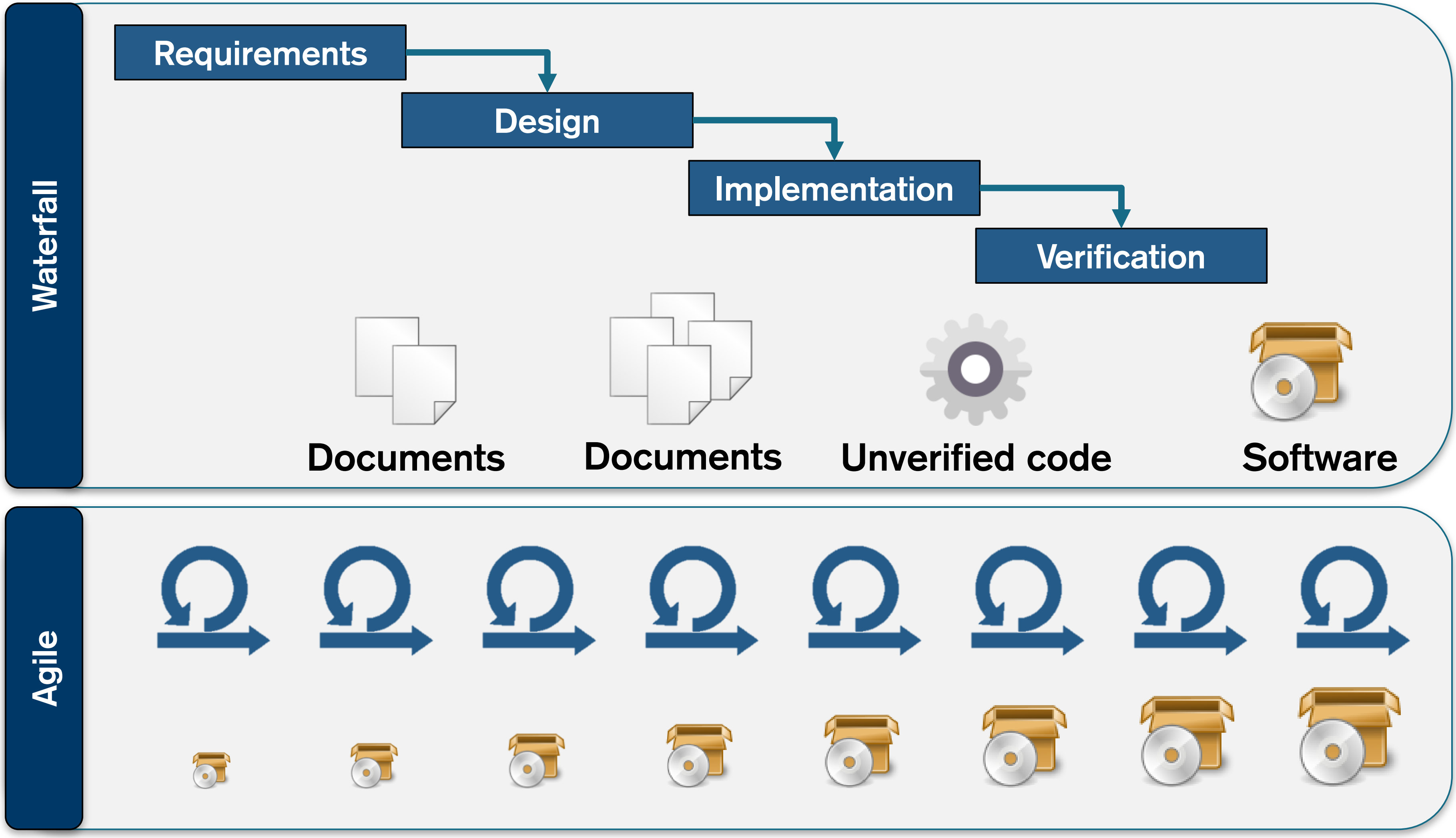


Develop on cadence
Deliver on demand

Highest business value first

Feedback source

Agile // Incremental Value Delivery



Waterfall

Requirements

Design

Implementation

Verification

Documents

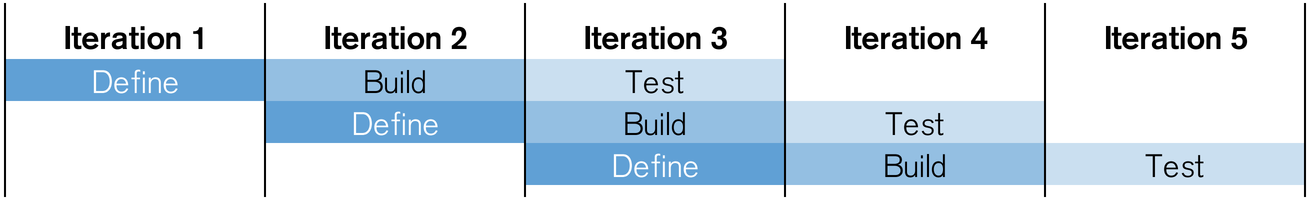
Documents

Unverified code

Software

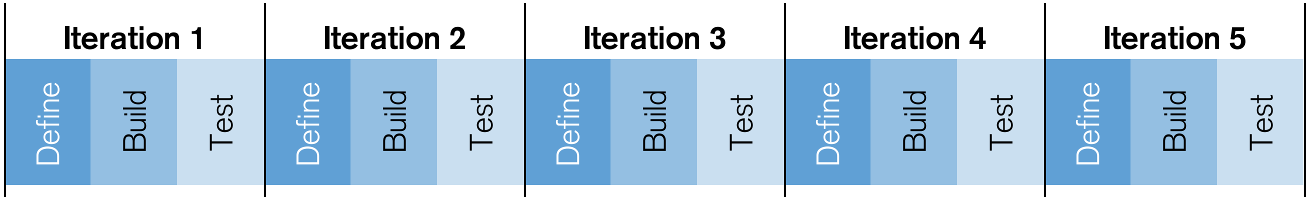
Agile

Agile // Waterfalling Iterations



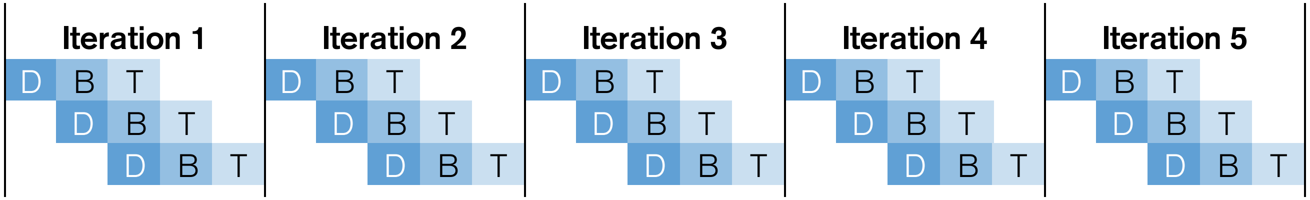
Iteration 1	Iteration 2	Iteration 3	Iteration 4	Iteration 5
Define	Build	Test		
	Define	Build	Test	
		Define	Build	Test

✗



Iteration 1			Iteration 2			Iteration 3			Iteration 4			Iteration 5		
Define	Build	Test	Define	Build	Test	Define	Build	Test	Define	Build	Test	Define	Build	Test

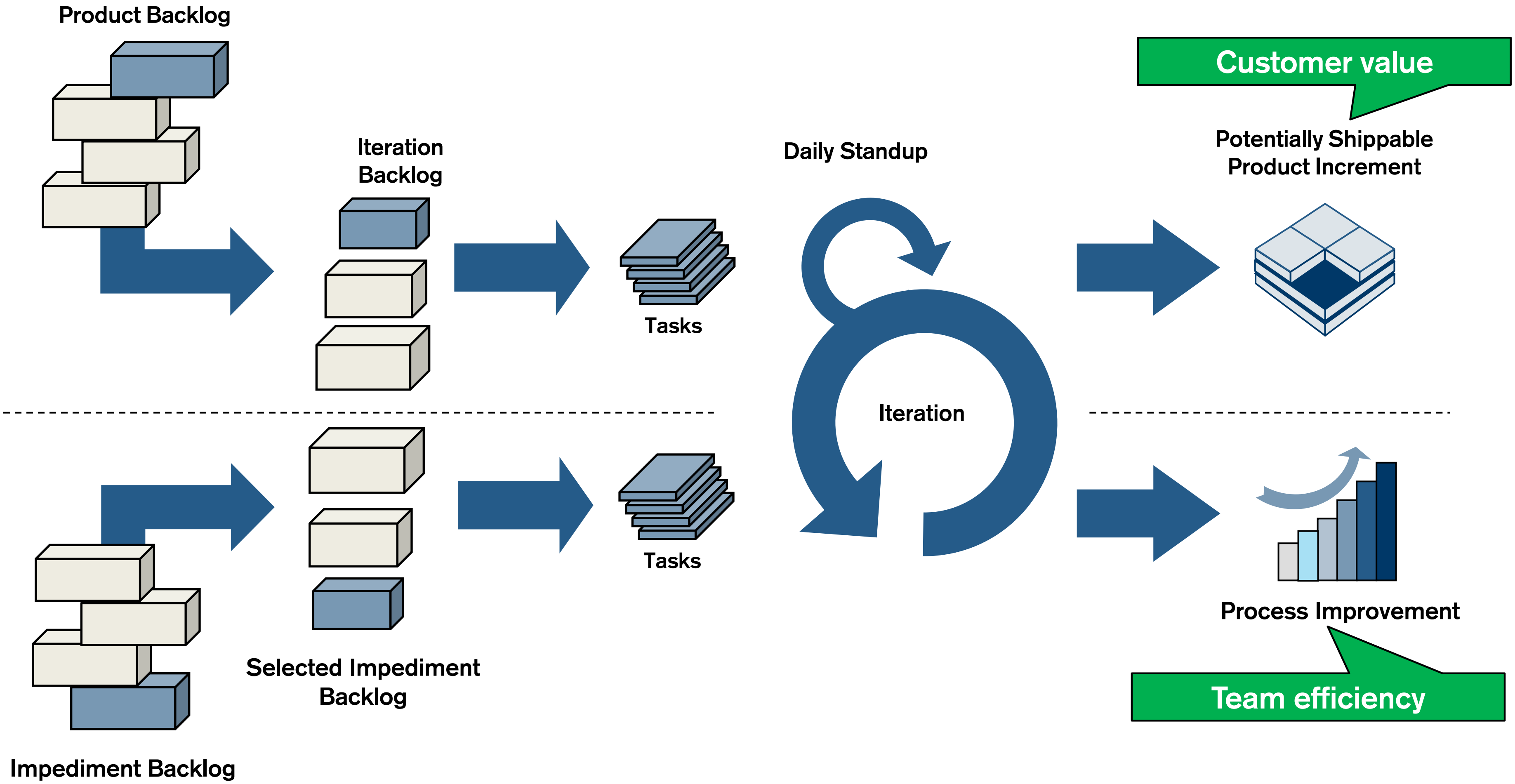
✗



Iteration 1					Iteration 2					Iteration 3					Iteration 4					Iteration 5				
D	B	T			D	B	T			D	B	T			D	B	T			D	B	T		
	D	B	T			D	B	T			D	B	T			D	B	T			D	B	T	
		D	B	T			D	B	T			D	B	T			D	B	T			D	B	T

✓

Agile // Scrum complete cycle



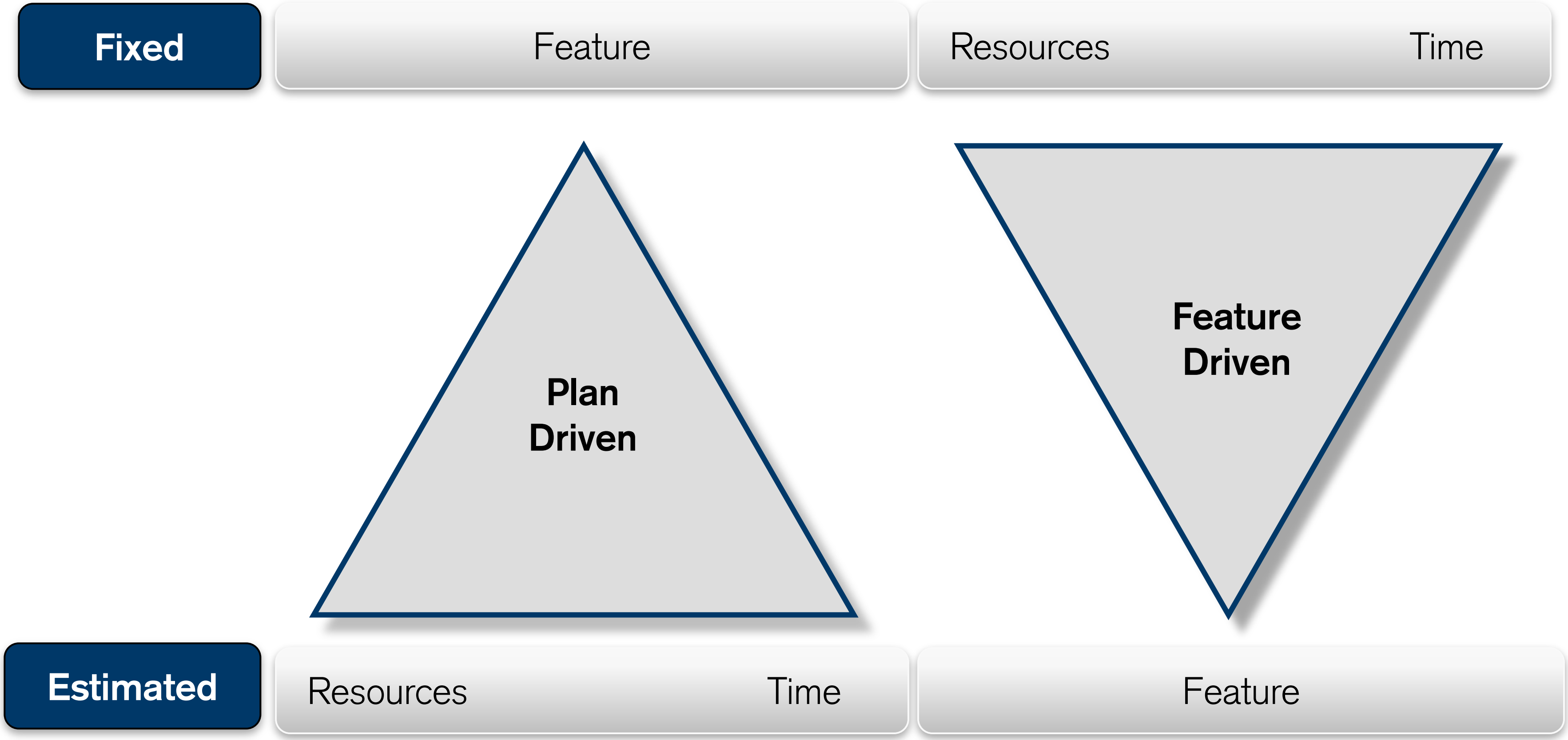
Agile // Scrum ceremonies

- **Iteration Planning**
  Select how many items (Stories) from the Product Backlog (top of the backlog) will be delivered in the iteration (and go in the Iteration Backlog)
- **Daily standup**
  Daily synchronization meeting between the team members in order to see the progress of the iteration
- **Iteration Review & Demo**
  End of iteration meeting to show the stories that have been achieved (to the Product Owner)
- **Iteration Retrospective**
  Workshop where the good and bad parts of the previous iteration are analyzed in order to improve the next one and be more efficient
- **(Backlog Grooming)**
  Refinement of the Product Backlog, pre-selection of a set of stories and presentation to the Team

Agile // Super Scrum Master

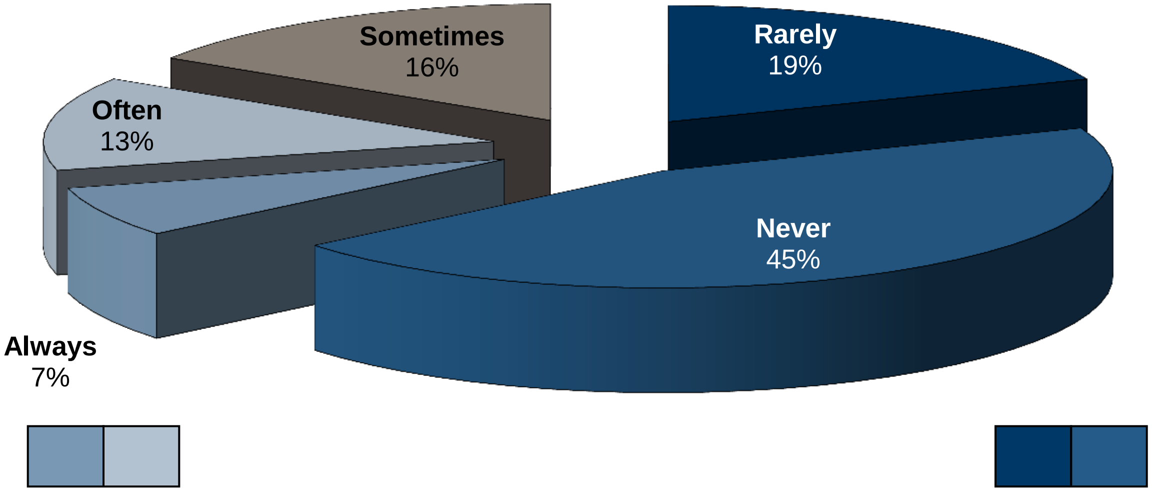
How Agile is better

Improve // Inverting Iron Triangle



Fixed	Feature	Resources Time
Estimated	Resources Time	Feature

Improve // Features used (classical)



Always or often used

20%

Never or Rarely Used

64%

Standish Group Study Reported at XP2002 by Jim Johnson. Chairman

Improve // Features used (classical)

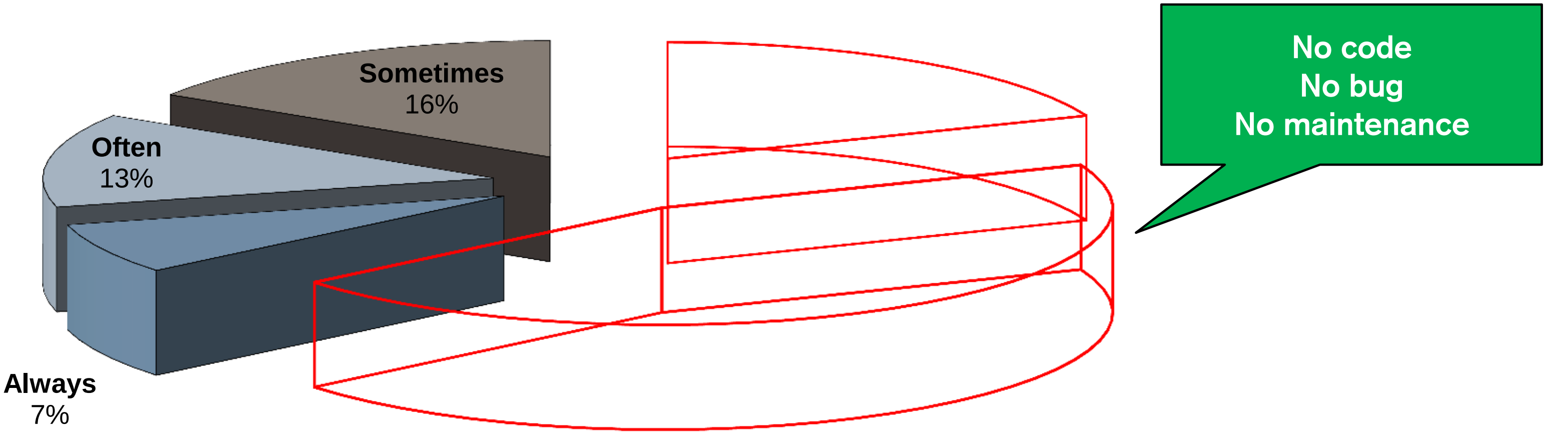
64%

Of the delivered features are never or rarely used

36%

Are used sometimes or more

Improve // Features used (Agile)



No code
No bug
No maintenance

Standish Group Study Reported at XP2002 by Jim Johnson. Chairman

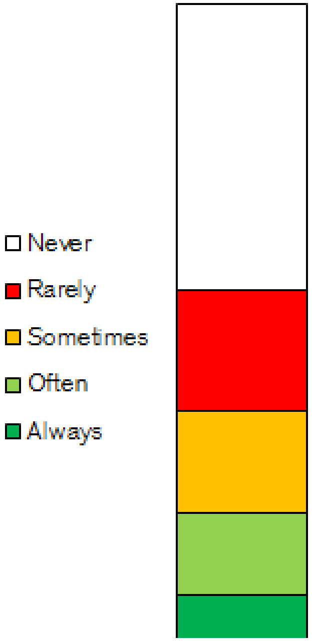
Improve // Cost of feature used

In a 4 mio USD project, 2.6 mio are spent on rarely or never used features

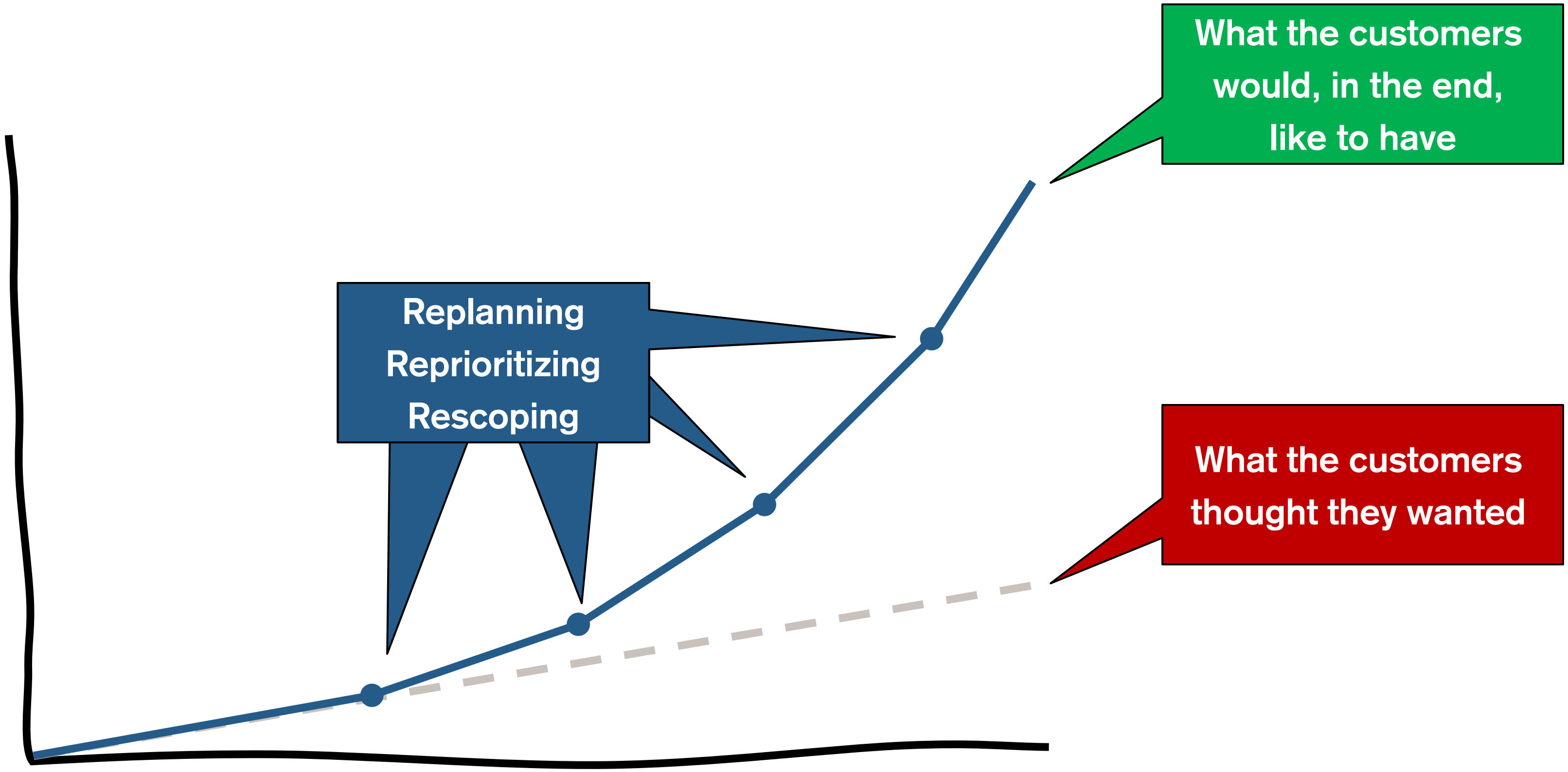
13 bugs out of 20 I solved this week are due to rarely or never used features

Improve // EMA

- Imaging a building with 28 floors, like EMA
  - Floors 1-2 would always be used
  - Floors 3-5 would be often used
  - Floors 6-10 would be used sometimes
  - Floors 11-15 would be used rarely
  - Floors 16-28 would never be used

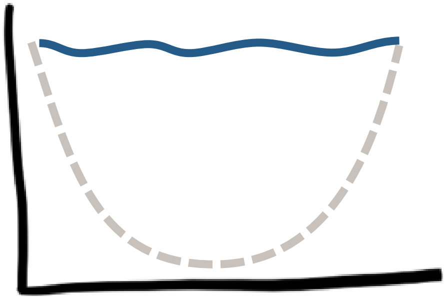


Improve // Agile Value Proposition

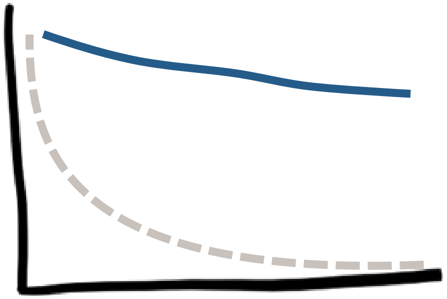


Agile
Waterfall

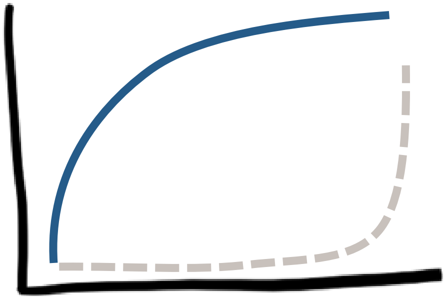
Improve // Agile Value Proposition



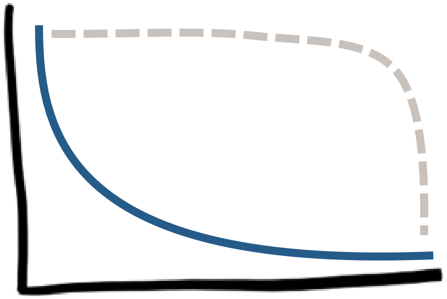
Visibility



Adaptability

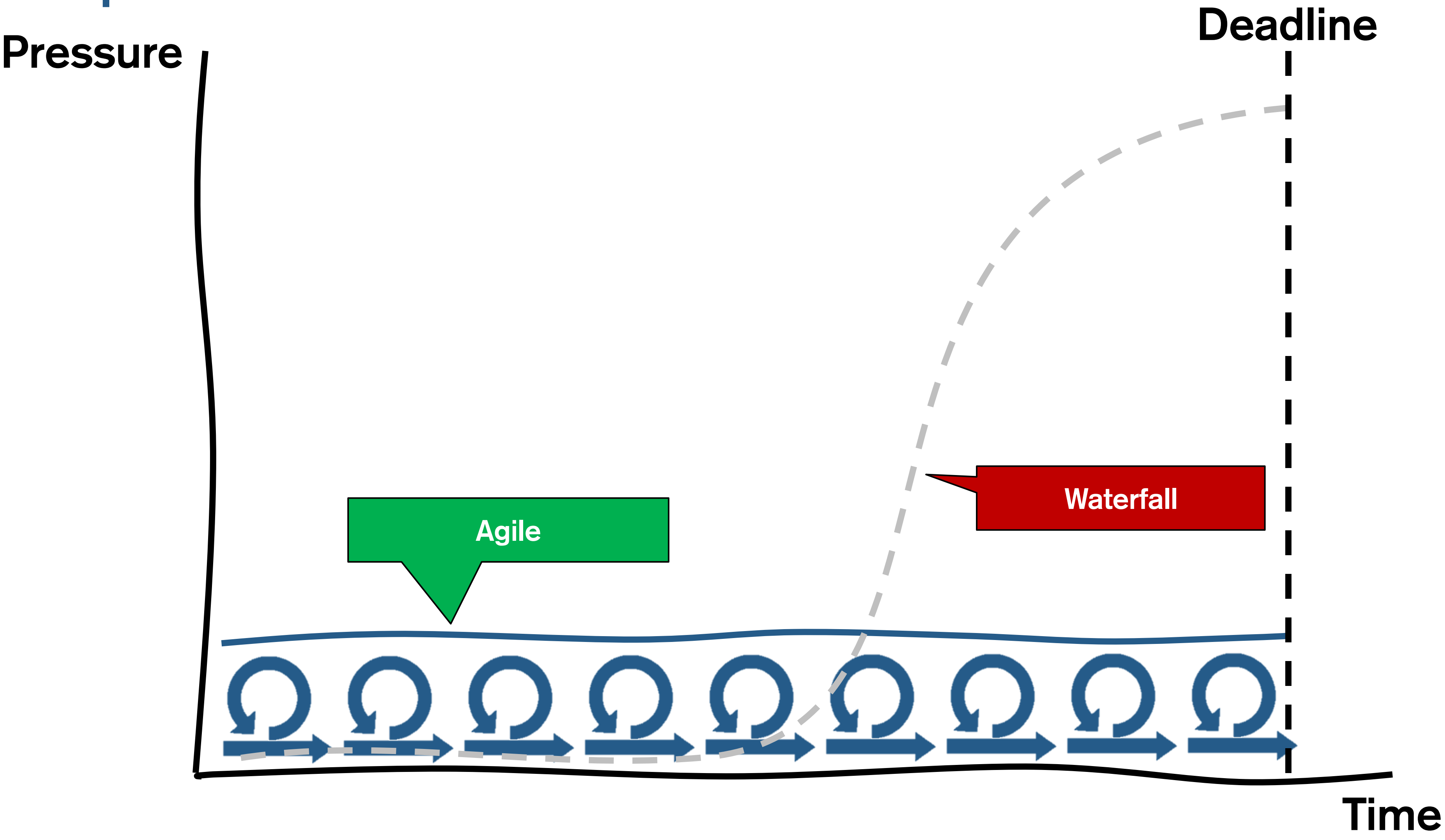


Business Value

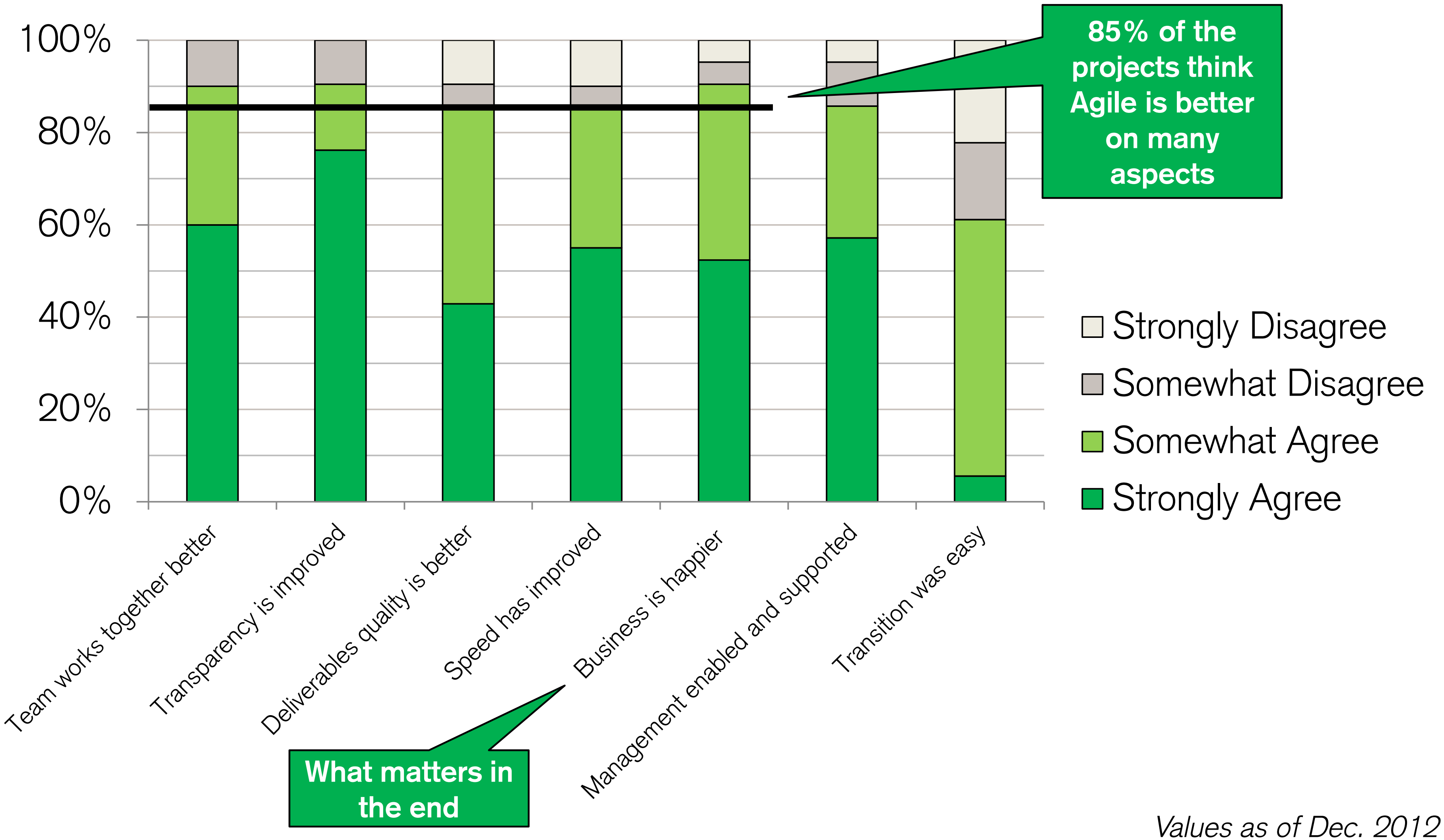


Risk

Improve // Death March



Improve // Actual impressions



85% of the projects think Agile is better on many aspects

What matters in the end

Values as of Dec. 2012

Ball Point Game

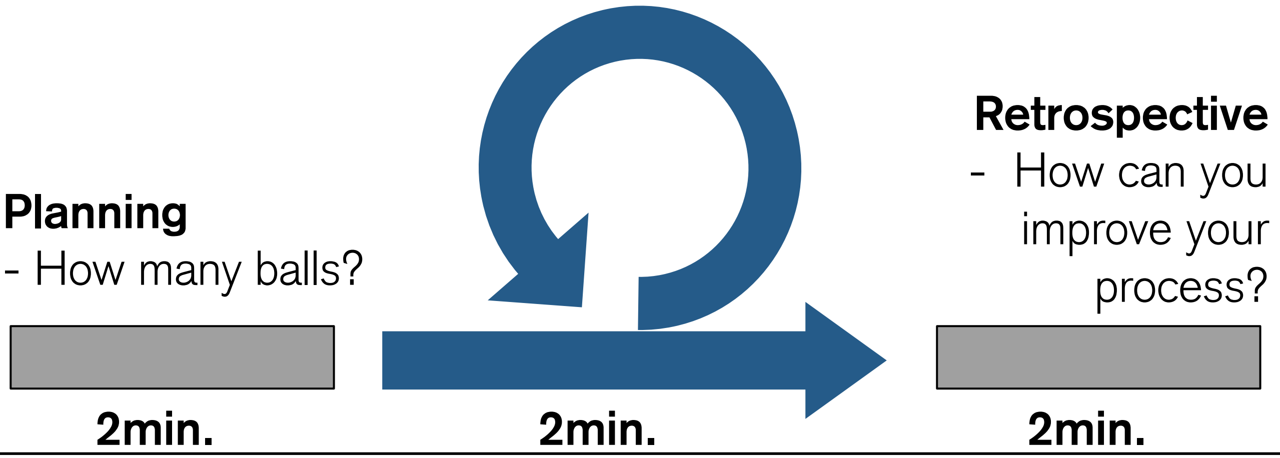
Exercise – Instructions

Goal

Pass as many balls as possible through every team member in 2 minutes. The team gets a point for each ball passed through every member of the team where the first person to touch that ball is also the last.

Rules

- You are two big team
- Ball must have air-time
- No ball to your direct neighbor
- Start Point = End Point
- 1 Ball at a time per person


Planning

- How many balls?

2min.

2min.

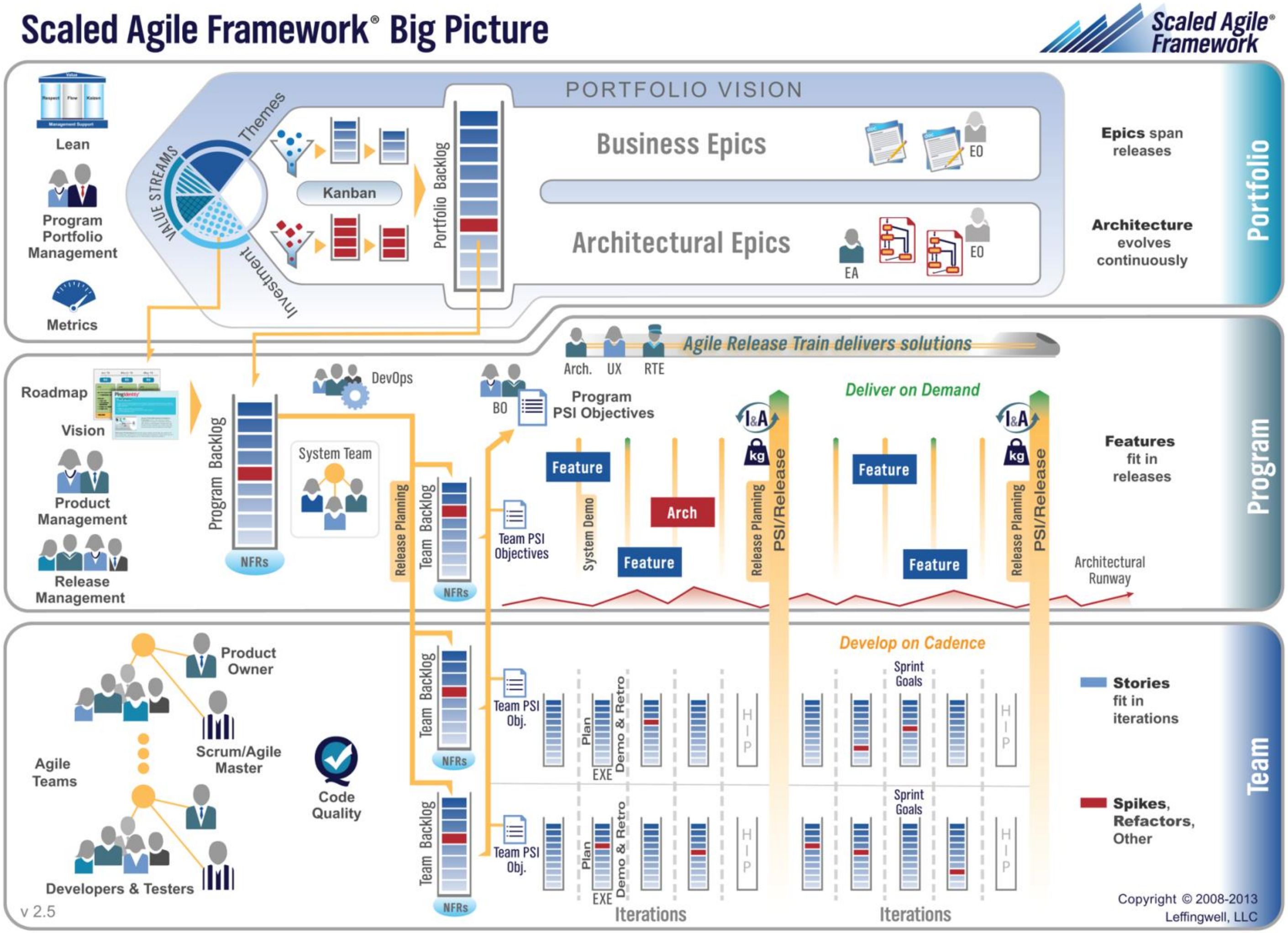
Retrospective

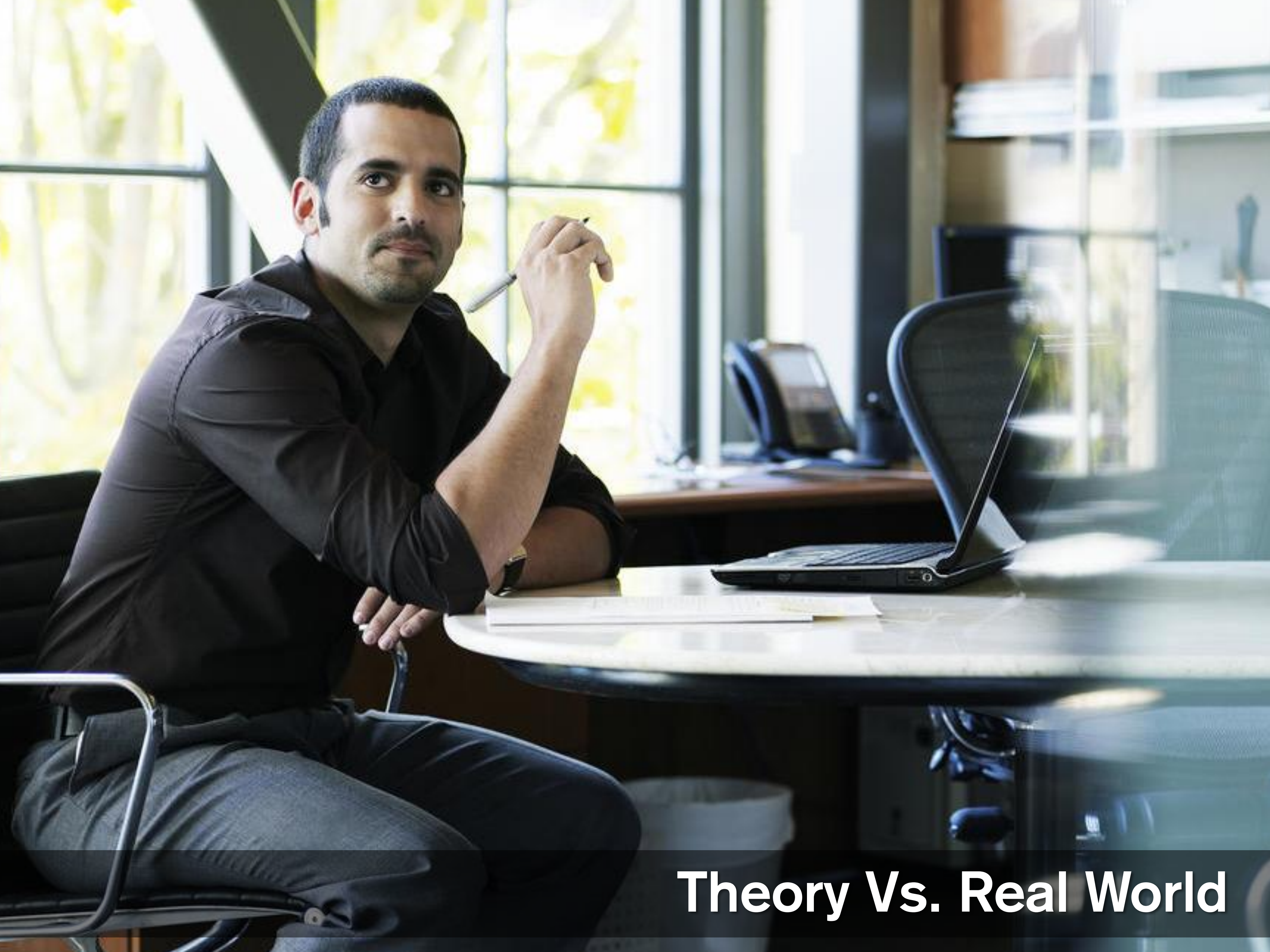
- How can you improve your process?

2min.

How does Agile fit in a big company

Big company // Scaling





Theory Vs. Real World

Real world // Various Facts

- Huge differences of knowledge between teams
  - SVN, CI, TDD

- Teams around the globe
  - Create independent teams, Proxy PO

- Trust
  - Between Team <> PM
  - Between Team <> Customer

- Silos
  - I care only about the coding, testing is a tester issue

Real world // Various Facts

- 30% team members immediate buy-in, 60% after some time, 10% never
  - The same applies to projects

- Some are interested only in the good parts (for them)
  - Both on management and team side

- Difficult to get away from command and control way
  - Difficult time letting people choose teams, choose roles
  - Assign work instead of letting teams self-organize

- Team (self) **discipline** and role selection is key

Real world // Various Excuses

- It doesn't work for big projects → Scaled Agile
- Doesn't work in a bank, we need quality → Quality is better (more tested)
- What about QA, they cannot do their work in 2 weeks → Team should deliver code without bugs, QA just verification, not detection
- Business/customers will never like that → Oh yes they will

Real world // Funny stories

- Story points estimation, really ?

- We can’t deliver every two weeks, we are not at Google here

- Scrum Master time allocation
  - Do I really have to facilitate all those meetings ? I am Scrum Master for 6 teams + BA
  - We cannot have afford a (100%) Scrum Master

Real world // Funny stories

- Yes I know Scrum
  - It doesn’t work for us but you can manage your own tasks with it
  - 5 minute retrospective since it doesn’t provide any value

- I need a WBS! What do you want to do with it? I just need a WBS

- We do not have any plan, we don’t know what we do next week since we are Agile

Real world // Some positive results

- ZH Scrum Master
  - From 20% achieved to 90%
  - Increasing motivation/focus/discipline
- Always more success stories
  - Always more projects willing to adopt
- Some project always want to go “deeper”
- Some people never want to go back
  - Better mood
  - Better performance
  - More trust and accountability

End