UNIVERSITY OF CAMBRIDGE

Agile: Collaboration vs Control

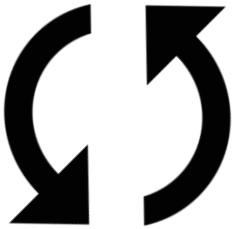
Dr. Nick Mattin,
Simon Redhead,
Dr. Sibel Allinson,

05/07/2017

Background

Do more with less

Maintain Highest Quality

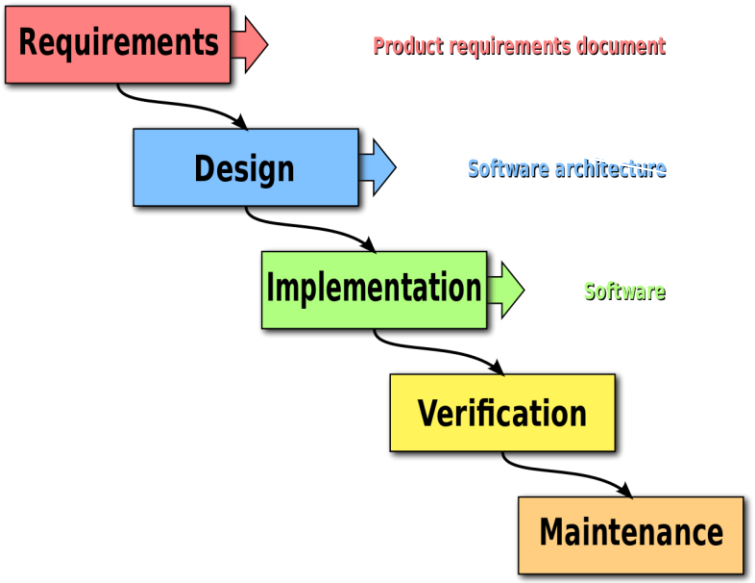


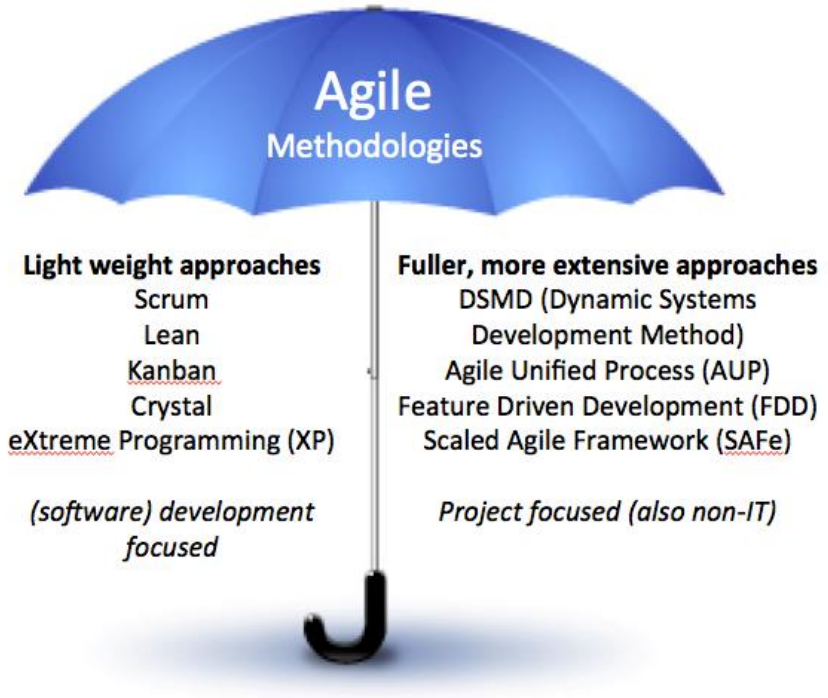
Faster Delivery

Reduce Project Risk

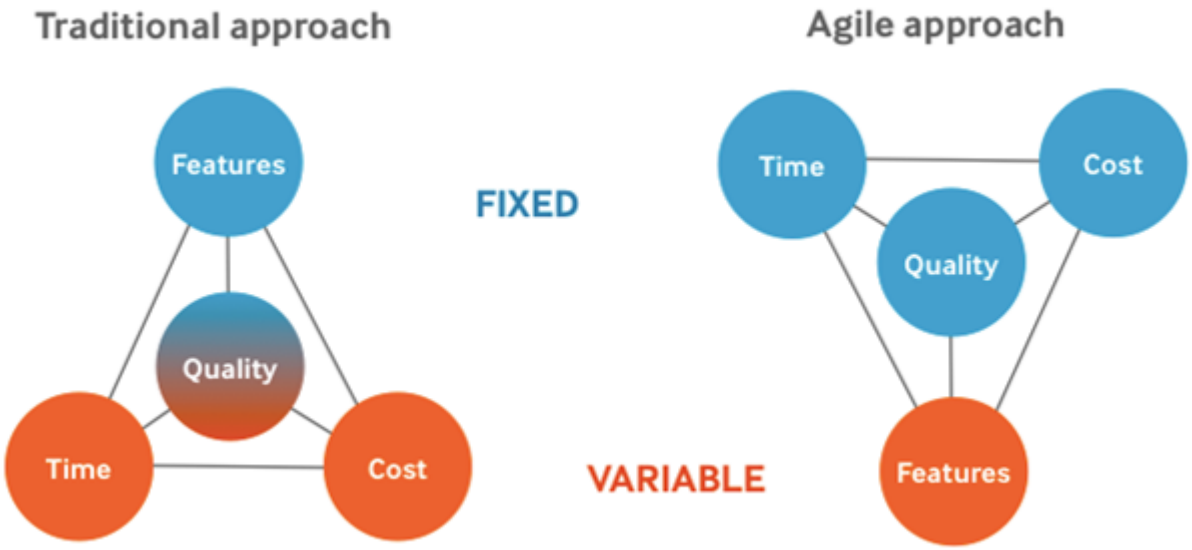
Environment

Multiple Approaches





Waterfall & Agile; Variables



Waterfall & Agile; Key Differentiators

Agile	Waterfall
Change oriented	Meticulously planned
All stages iterative	Fixed and separate stages
Short Feedback Loop	No feedback until test
Requirements evolve	Requirements fixed
Collaborative	Hand over between stages and teams
Constant research	Preliminary research
Efficient and timely risk management	Risk averse

UNIVERSITY OF CAMBRIDGE

Not a case study

But an assemblage of experiences.

Let's talk about real problems.

- I'm going to talk about things that we've seen that don't work very well, *and have been able to fix.*
- These cases are all drawn from the real world (well, Cambridge, anyway)
- A collection of experiences, observations, successes and challenges
- All have been encountered through our work here
- Everything is true

But!

- None of it applies everywhere or to everyone
- All put together risks appearing negative: it's not as bad as it sounds!
- Remember: **we have fixed** these problems - and have excellent working relationships!

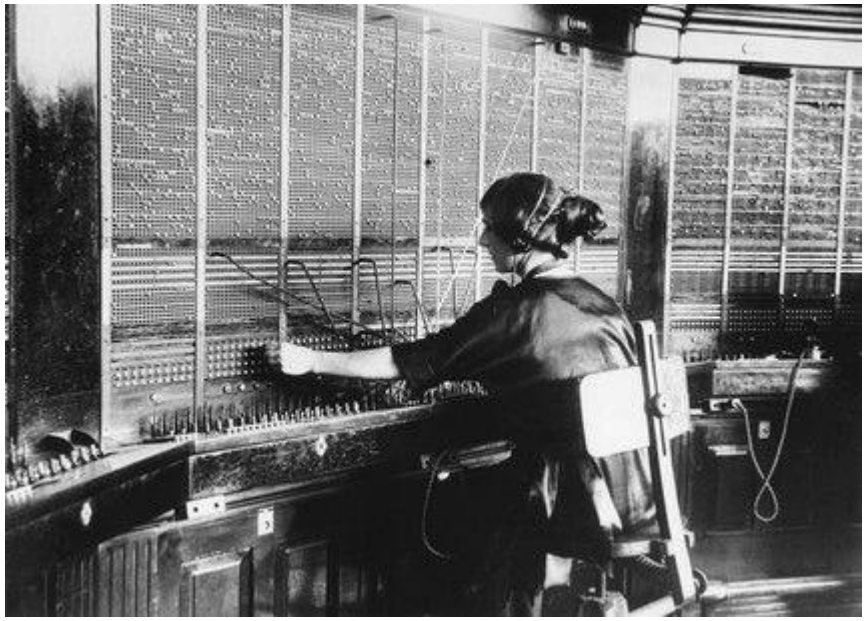
Things we have seen

Tick all that apply:

- ☐ Lots of information spread everywhere (data, systems, business knowledge)
- ☐ High workload, not enough staff
- ☐ Good at identifying problems, hard to implement right solutions
- ☐ No unified vision; differing perspectives hard to unite
- ☐ Systems difficult to modify in response to new needs or new ideas
- ☐ A small number of key staff glue everything together with invisible knowledge and practices
- ☐ Staff sometimes constrained (by the organisation) to have only a partial view
- ☐ Processes fractured, possibly also inefficient
- ☐ A large knowledge gap between practical ‘on the ground’ activities and senior leadership

Wetware

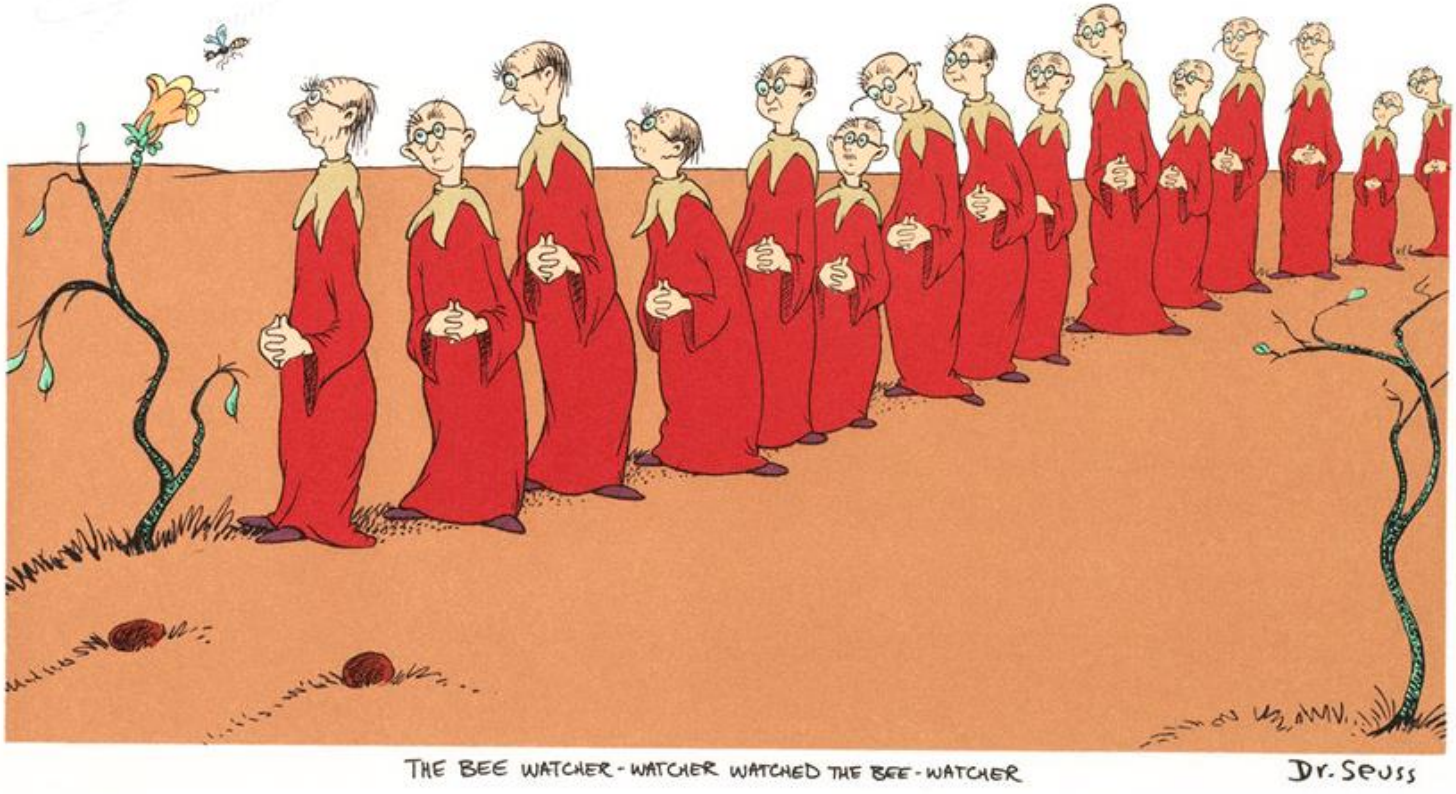
System incorporates a developer as a crucial component.



Left holding the baby

We provide the system, but end up being the only constant.

Micromanagement



It’s your fault!

Ever feel like someone’s lost something?

Like....

.....a sense of ownership?

- “It’s *your* system, *you* fix it!”
- “We don’t have time to test *your* system!”
- “Why is it taking *you* so long?”
- “Why didn’t you foresee `<some_deficiency>`?“
- “Tell me what the developers were working on yesterday.”

This is not the natural state of things - 1

Antipatterns have many possible undesirable outcomes, such as:

- Can only make small, tactical improvements (powerless to improve the ‘big picture’)
- Have a grand vision, but no way to bring it into being (∴ frustration)
- We are asked to implement a pre-formed solution to a problem that is barely understood (can mean expensive, ill-fitting, late – and, ultimately – disliked products)

This is not the natural (or only) state of things!

These outcomes all derive from learned behaviour.

People do what they know

....but can’t always see why it isn’t working as it should.

Control *vs.* Collaboration

- When outcomes aren't as expected, a natural tendency is to assert more control,
  but top-down, asserted control is often reactive, constraining.

Software is a creative activity – this is not the army!

- Overall control needs to exist, but it can also exist collectively
  We need to demonstrate that this is true

In the University, achieving this is always more about people, less about processes

- That's why we insist on involving stakeholders deeply in the solution development
- That's why we insist on talking continuously
- That's why we insist on showing everybody what we're doing frequently

The wrong kind of control

Control **and** collaboration

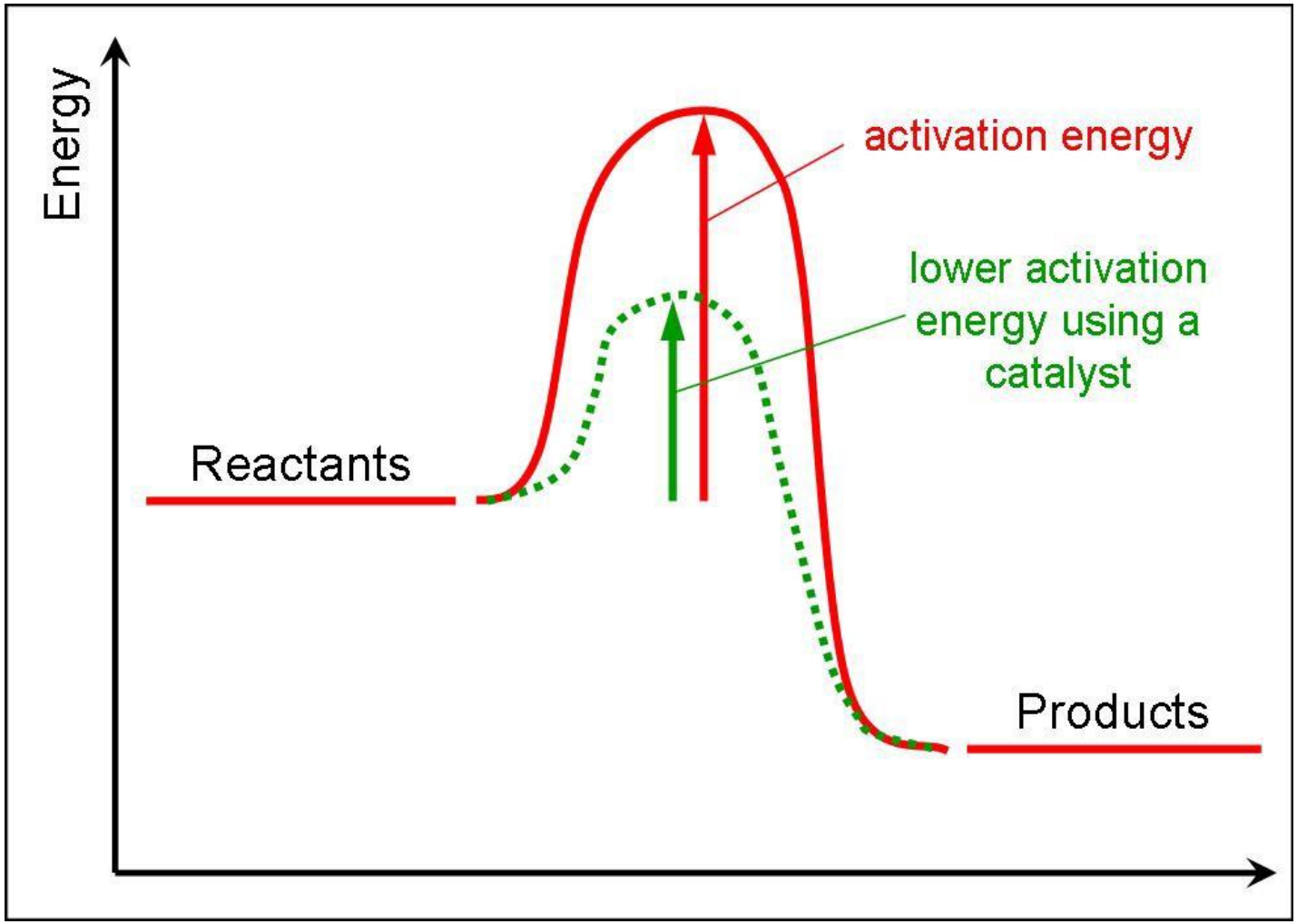
This is not the natural state of things - 2

We simply need a catalyst to help us reach beyond...

...the proximate ...the short-term
...the preconceived

If we know where we are going, we can define a path.

(the path might be uphill at first)



The catalyst we selected...

DSDM AgilePM (née Atern)

- A comprehensive (thorough enough)
  but lightweight (can choose just the helpful parts)
  project management framework.

- Inspired by Scrum, we add this lightweight wrapper
- We found our own balance - every project is different.
- Provides stable reference points, helpful perspectives
- When talking ‘up’ and ‘out’, a framework is more persuasive than pure enthusiasm:

  ...it provides authority – and methodology – and reassurance.

How?

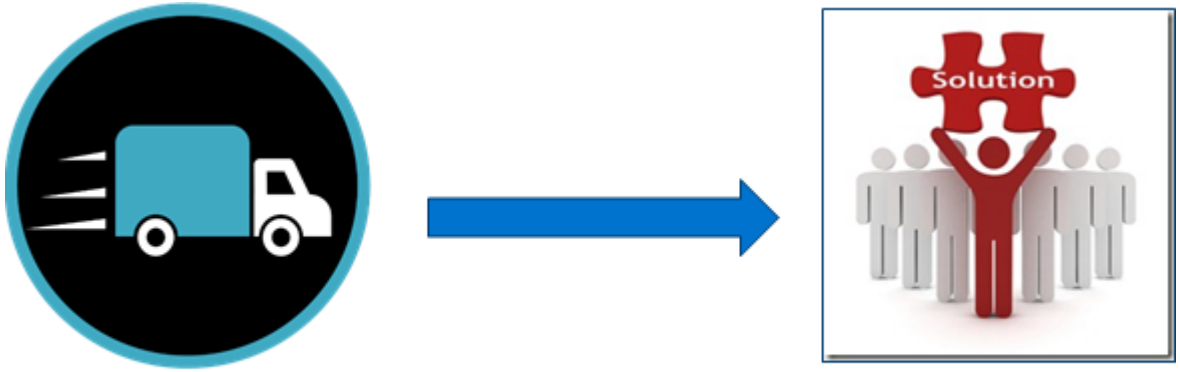
- Demonstrate effective decision making at every level.
- Provide tools and minimal structure to give confidence the current work is achievable.
- Focus on value at all times – and shed un-valuable activities (ruthlessly!) at all levels.
- **Show people the results.**

This demonstrates control.

It requires collaboration.

What is DSDM?

Dynamic Systems Development Method is an agile project delivery framework that delivers the right solution at the right time.



Adopted DSDM - Principles

1.Focus on the business need

2.Deliver on time

3. Collaborate

4.Never compromise quality

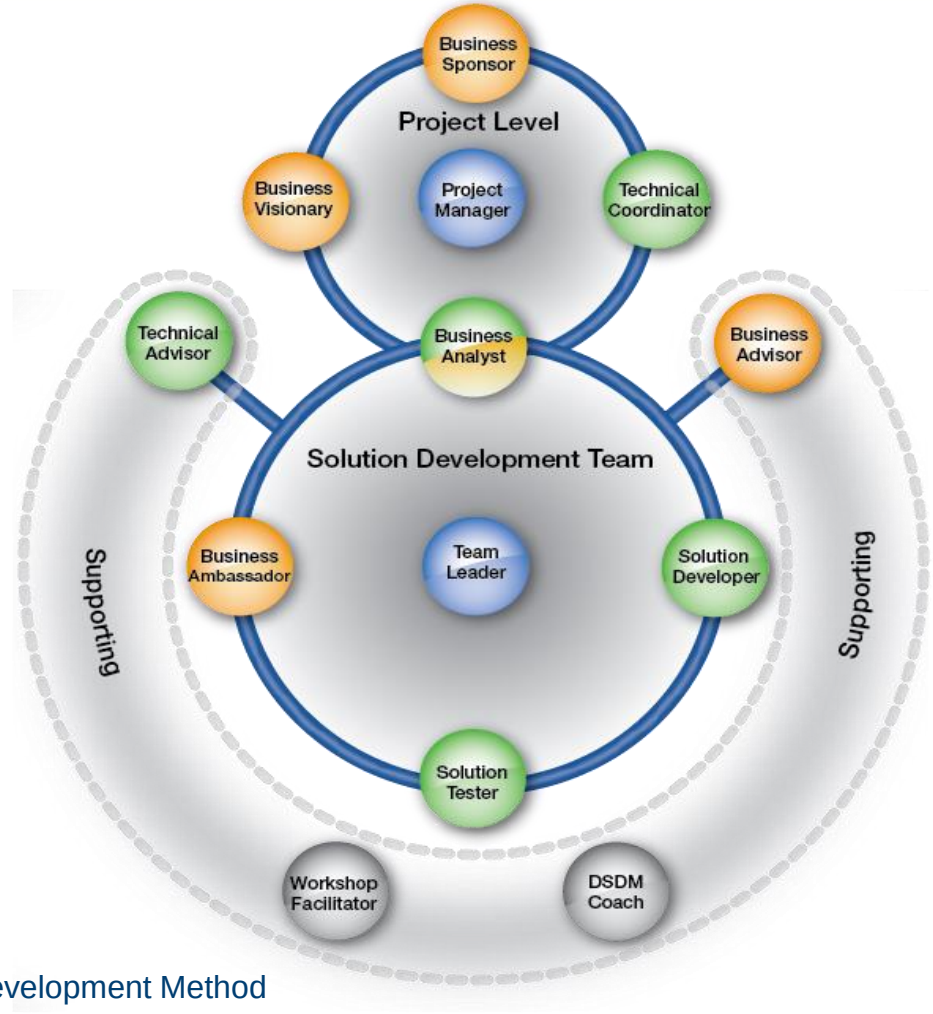
5.Build incrementally from firm foundations

6.Develop iteratively

7.Communicate continuously and clearly

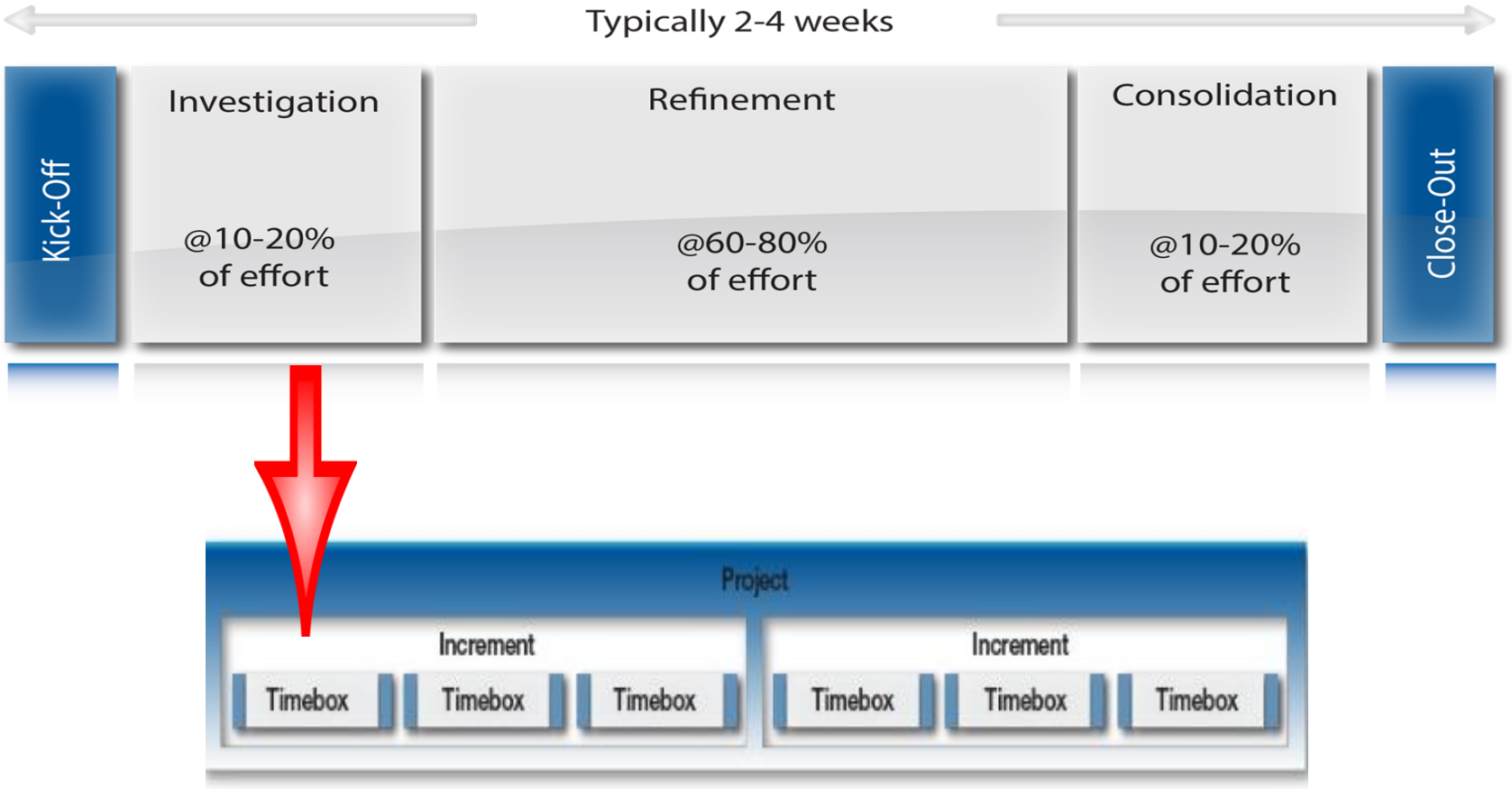
8.Demonstrate control

DSDM Team Model



DSDM= Dynamic Systems Development Method

Timeboxing



Timeboxes in the context of increments and project

Understanding and Managing Priorities : MoSCoW

80% of our work contributes to less than 20% of its value.

Must Have

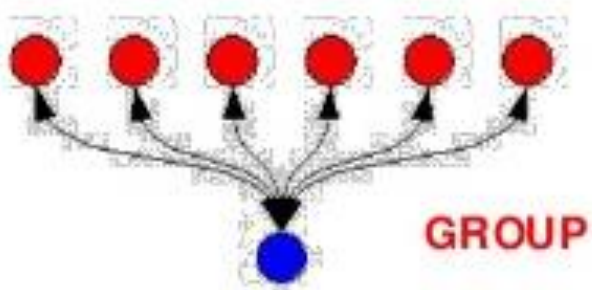
Should Have

Could Have

Won't Have this time

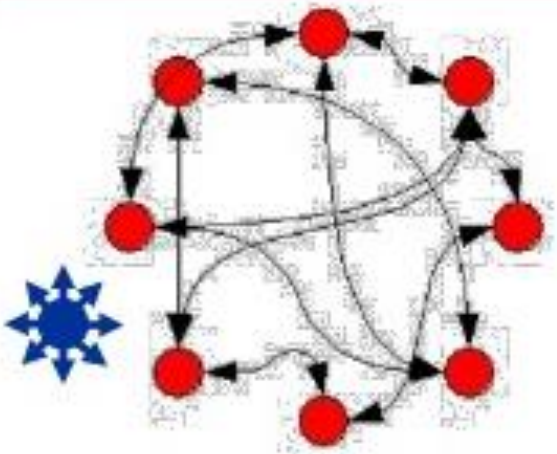
Collaboration vs Control

Traditional



Command & Control.
Micro-management.

Agile



Thinking together. Self-organization.
Management as servant leaders.

Questions and Answers


THANK YOU