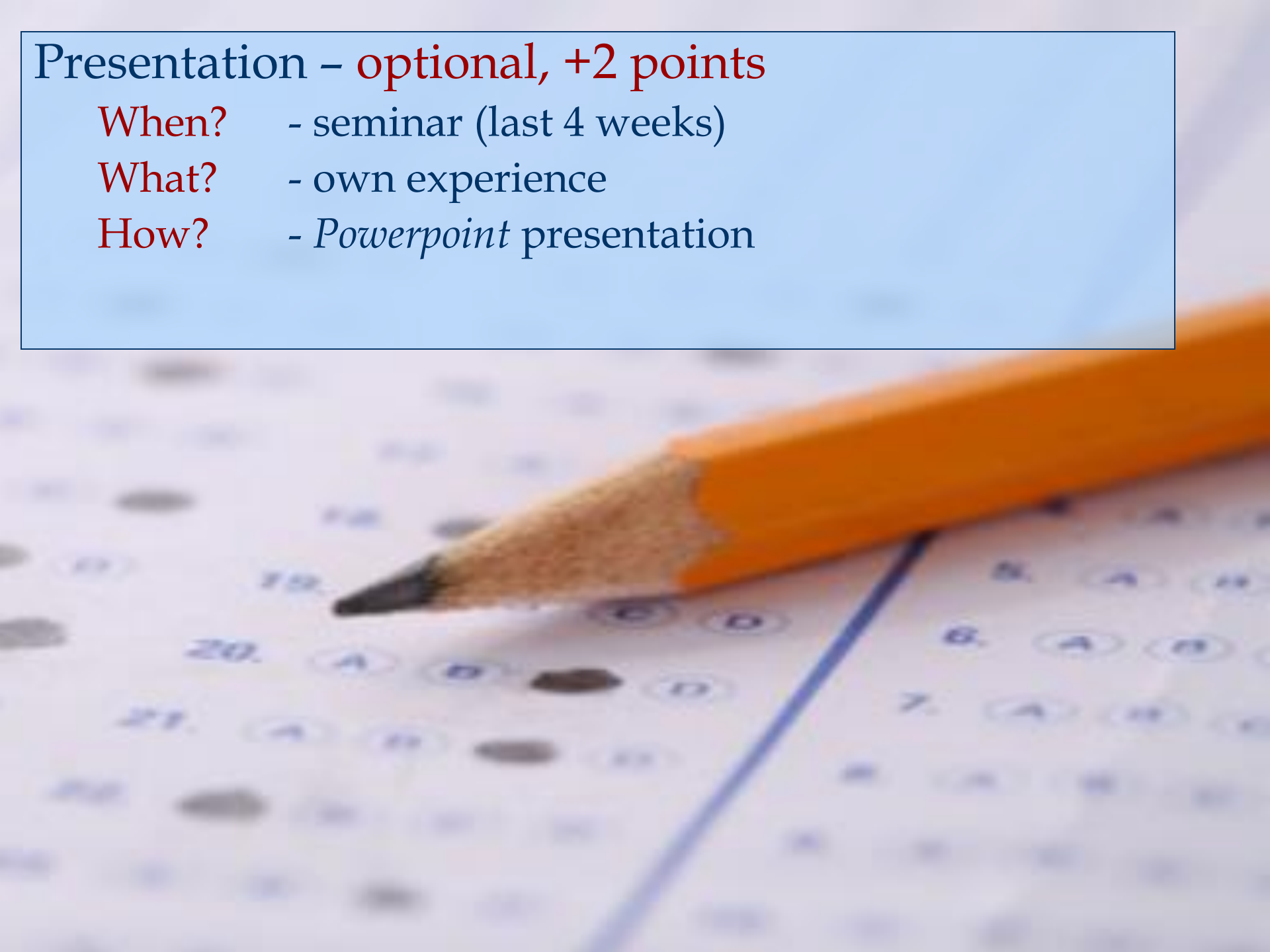
Introduction in Agile Methodologies

part 2

2

Course structure

- Agile Methodologies - Overview
- Scrum
- Extreme Programming
- Kanban
- Crystal
- The Agile Unified Process

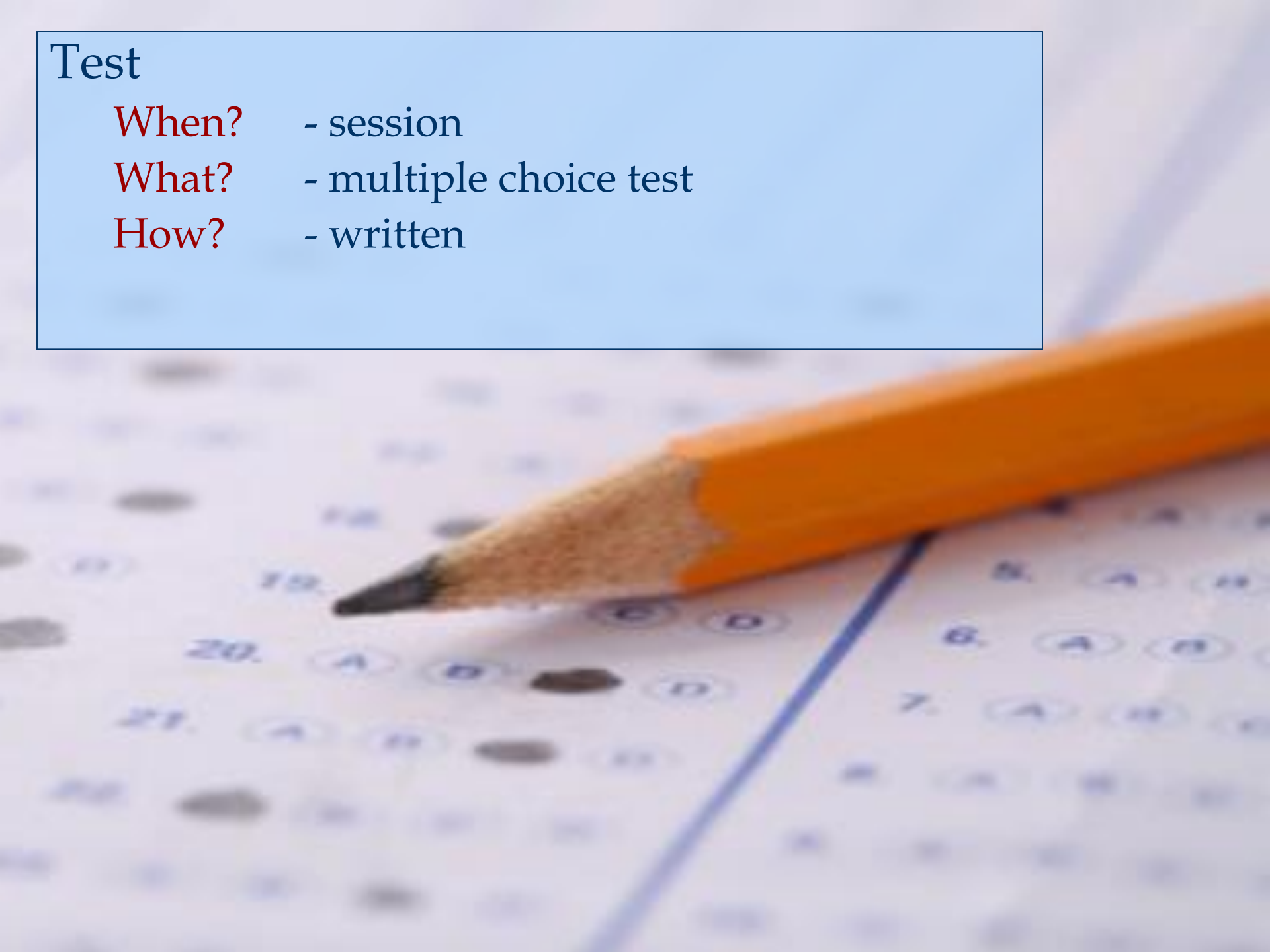


Presentation – optional, +2 points

When? - seminar (last 4 weeks)

What? - own experience

How? - *Powerpoint* presentation

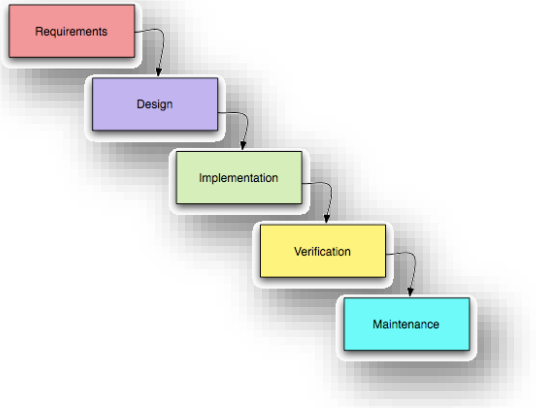


Test

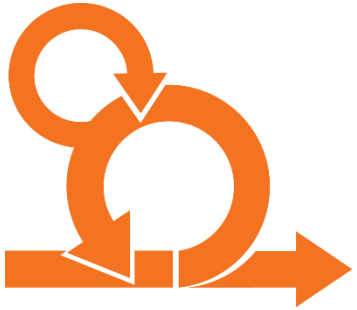
When? - session

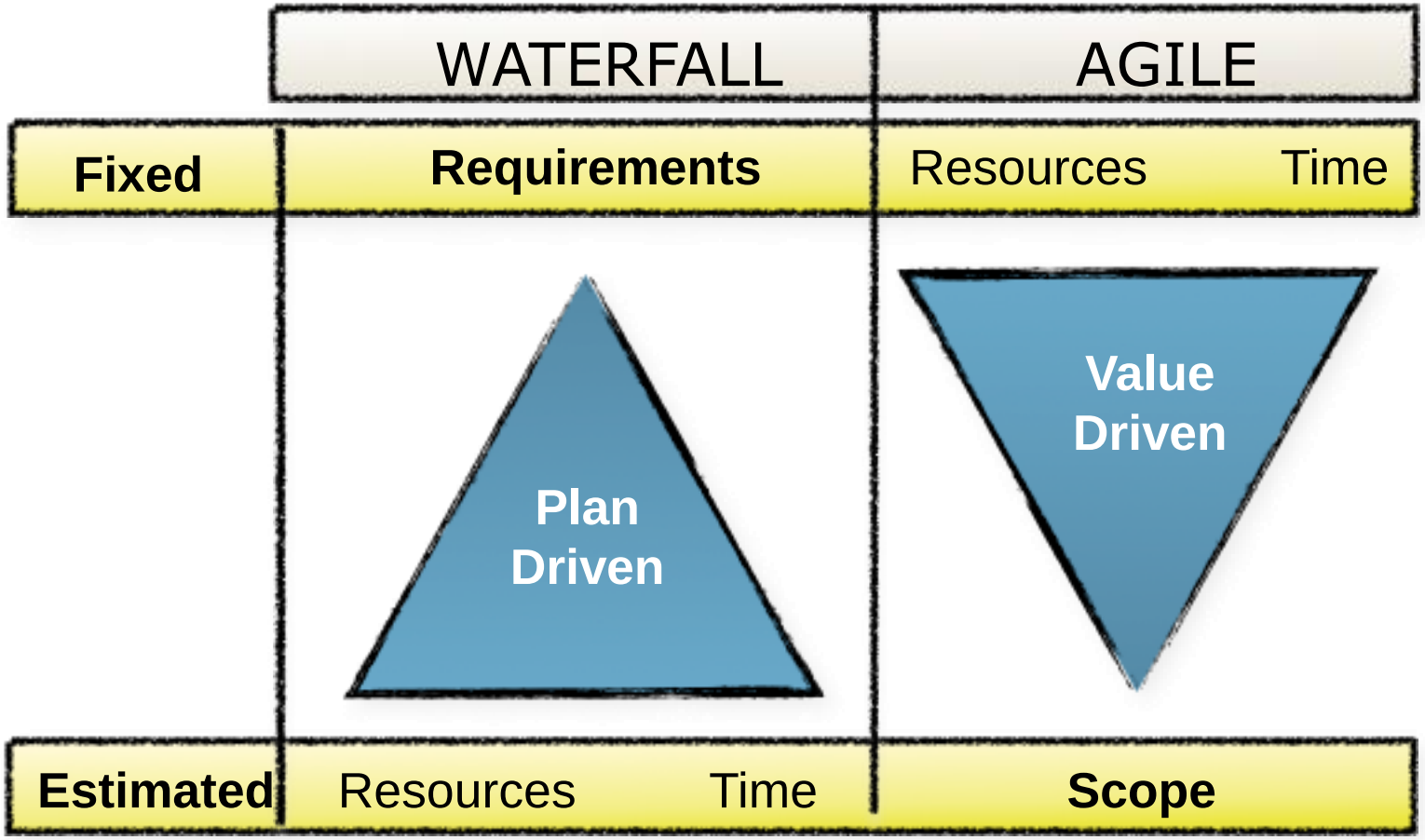
What? - multiple choice test

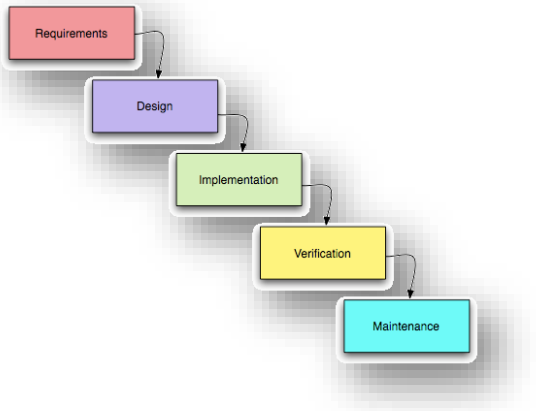
How? - written



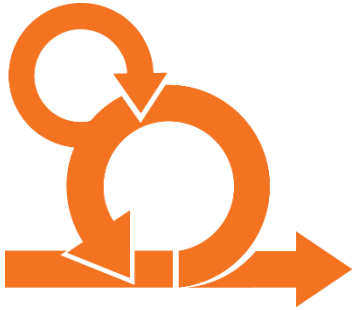
Overall picture




WATERFALL
AGILE
Fixed
Requirements
Resources
Time
Plan
Driven
Value
Driven
Estimated
Resources
Time
Scope


Requirements
Design
Implementation
Verification
Maintenance

Change

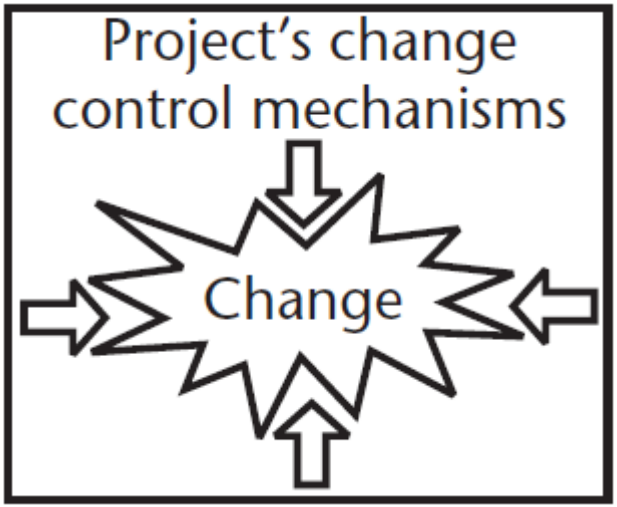


CHA
NGE

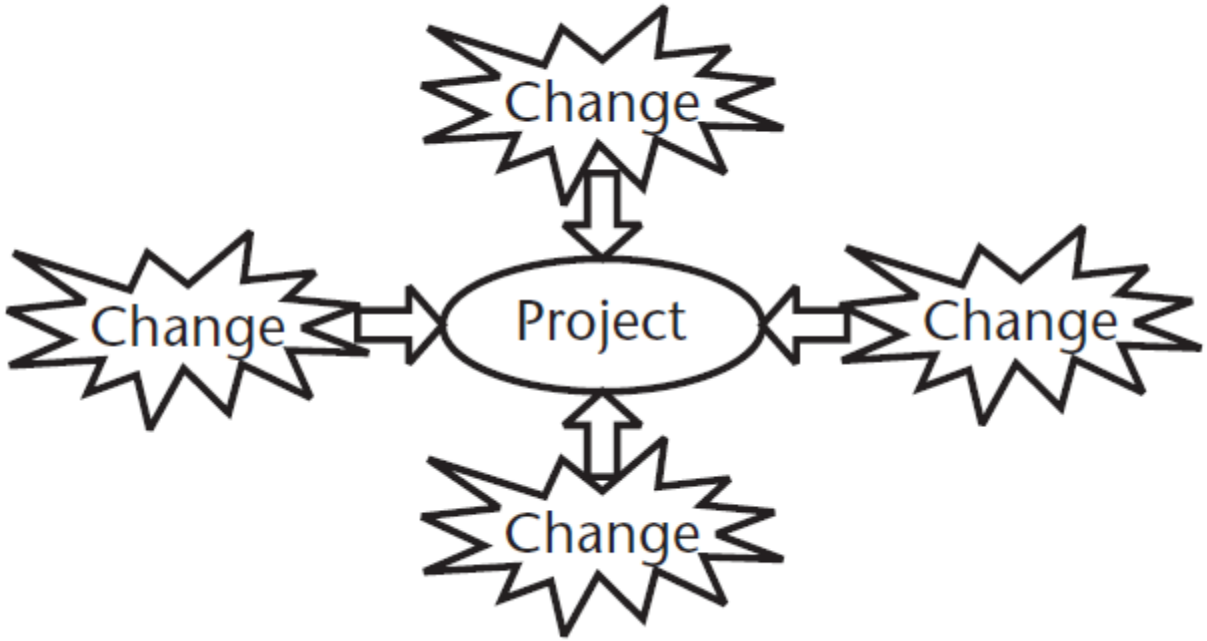
- Incomplete specifications

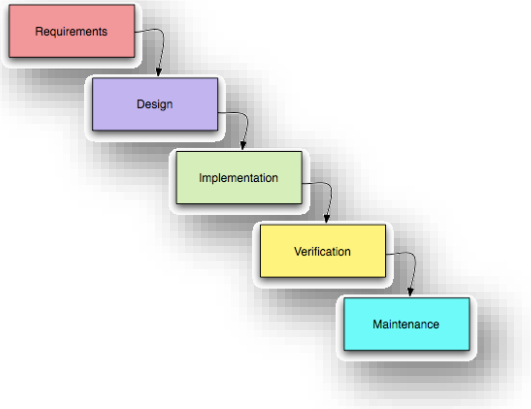
- Significant estimation errors

Controlling Change

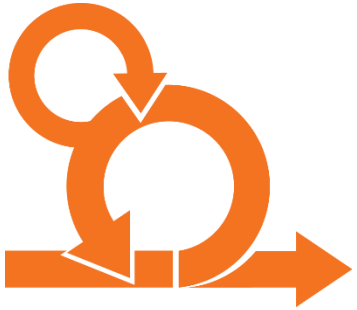


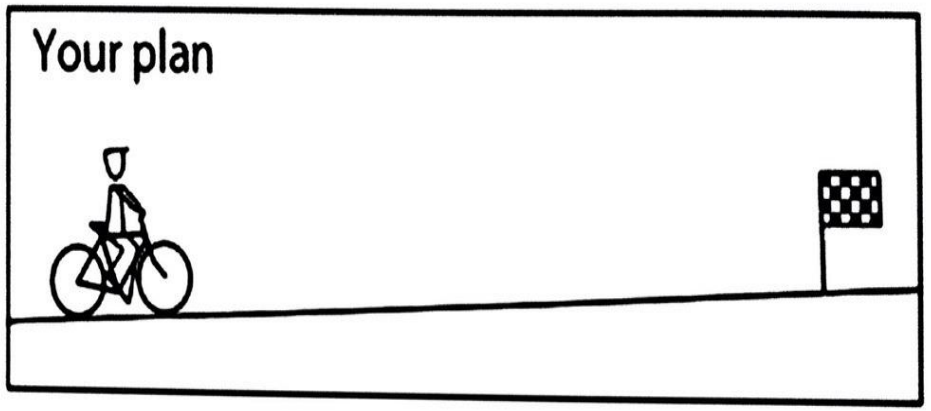
Reacting to Change

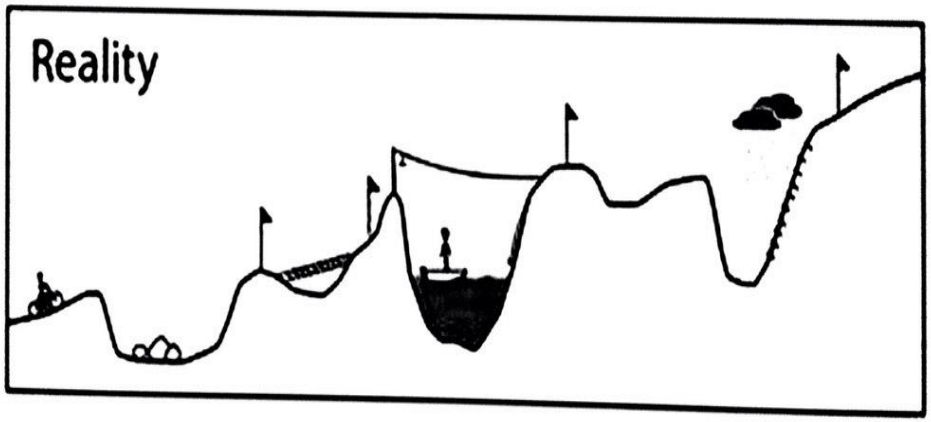



Requirements
Design
Implementation
Verification
Maintenance

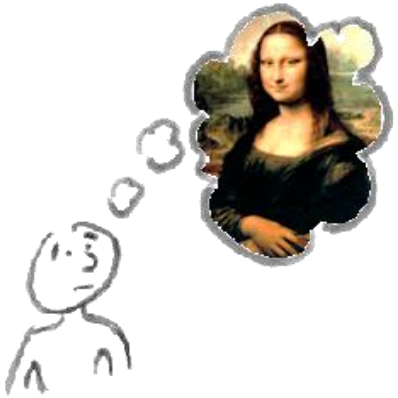
Planning




Your plan


Reality

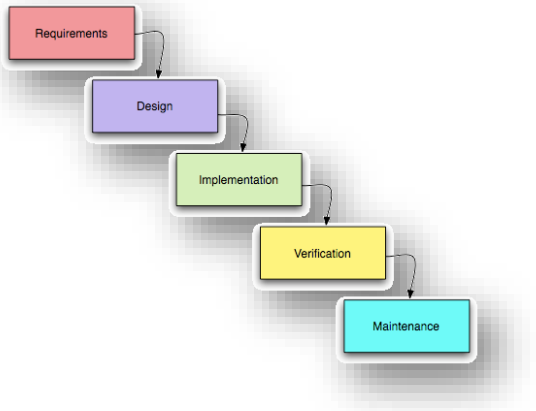
Plan-driven



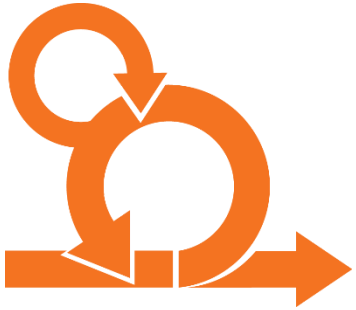
Evolutionary

Jeff Patton: http://www.agileproductdesign.com/blog/dont_know_what_i_want.html

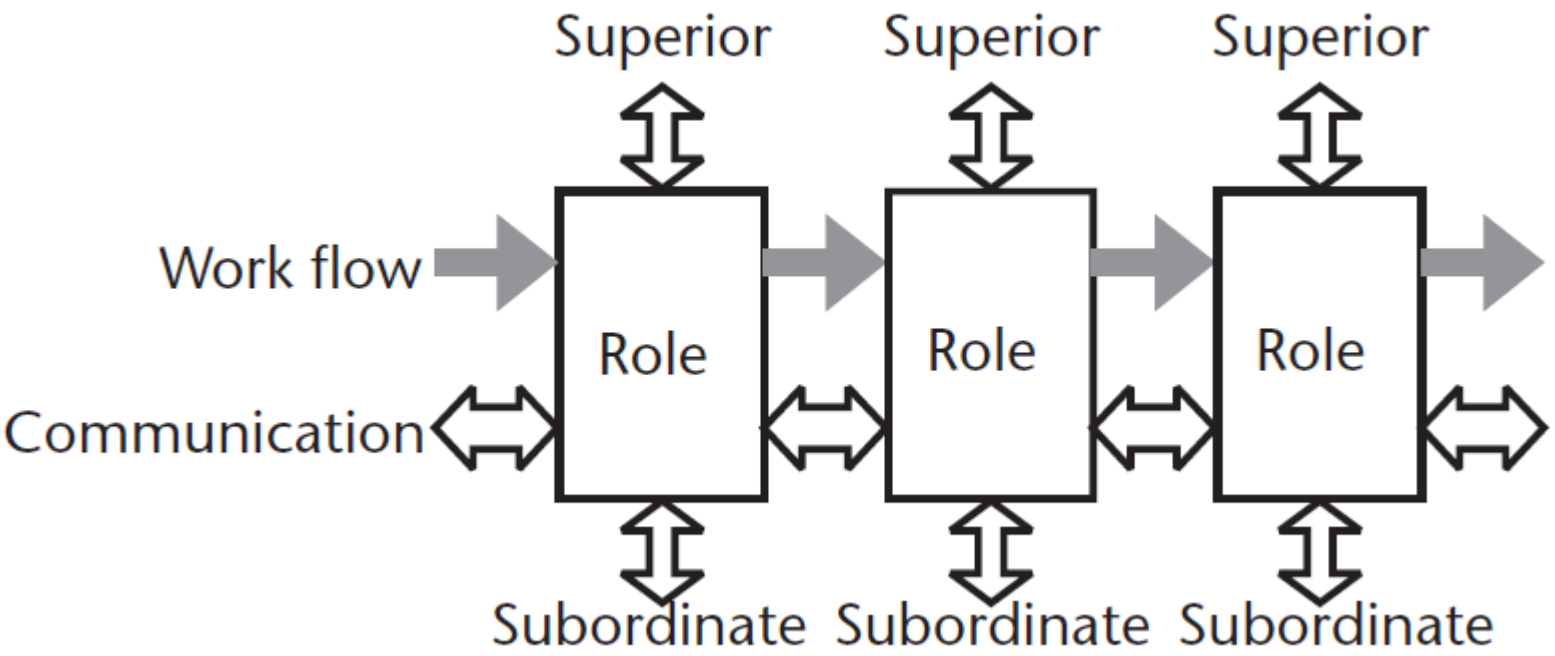
"Tahitians at rest" Paul Gauguin



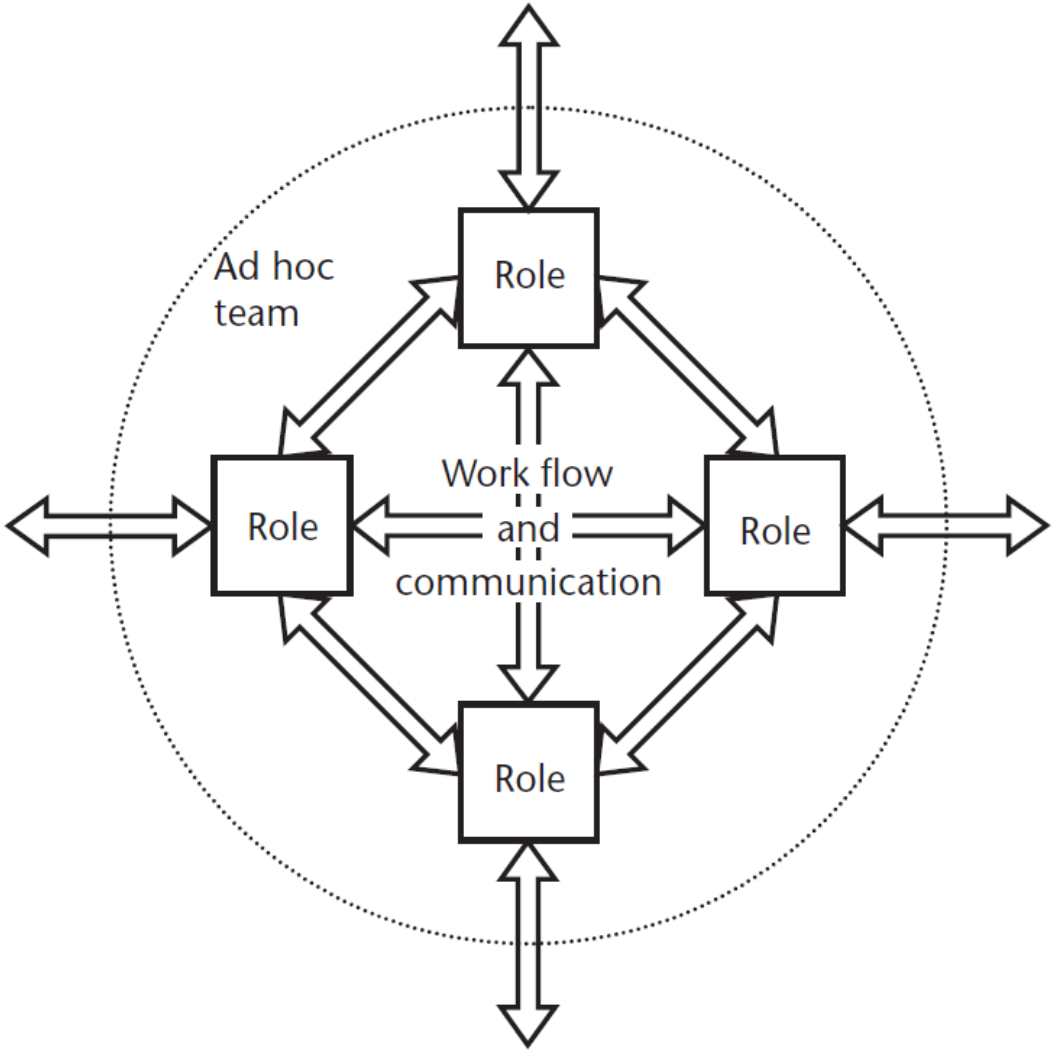
Communication

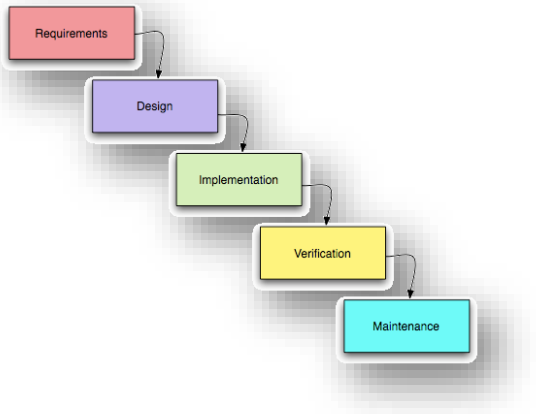


Hierarchical Organization

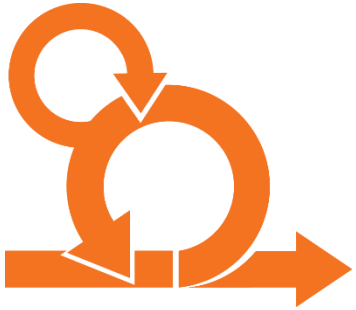


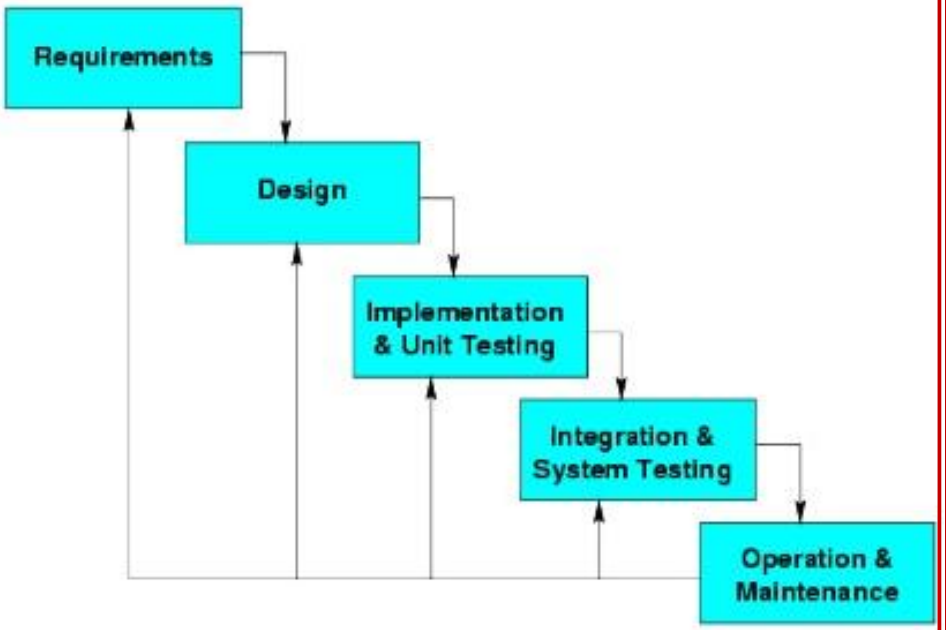
Cooperative Organization

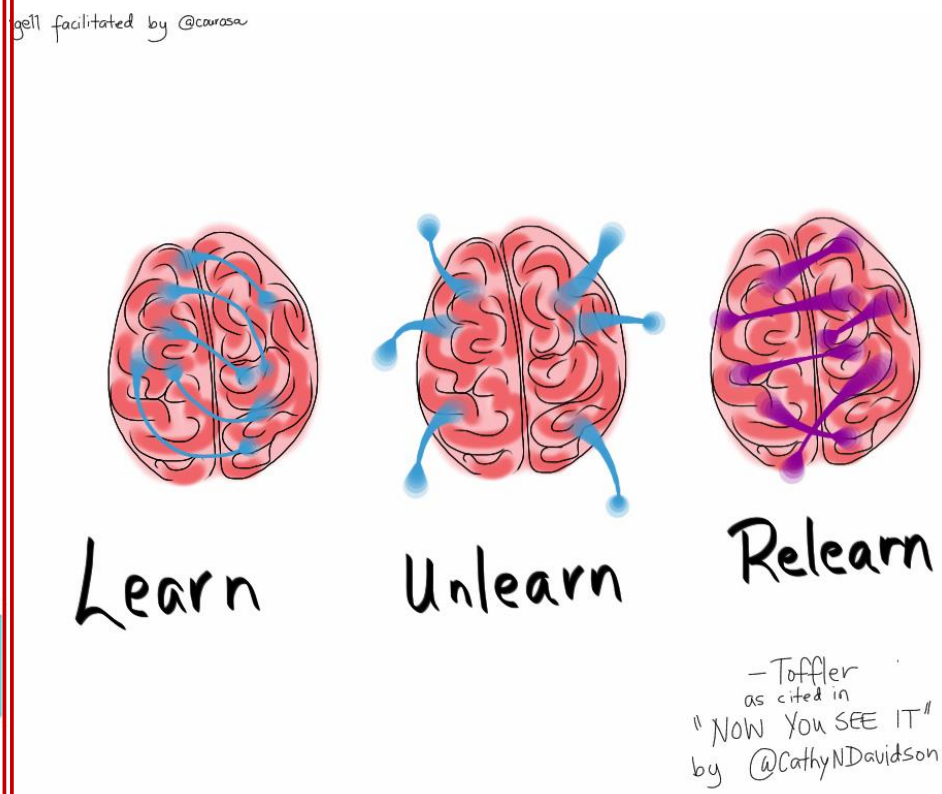



Requirements
Design
Implementation
Verification
Maintenance

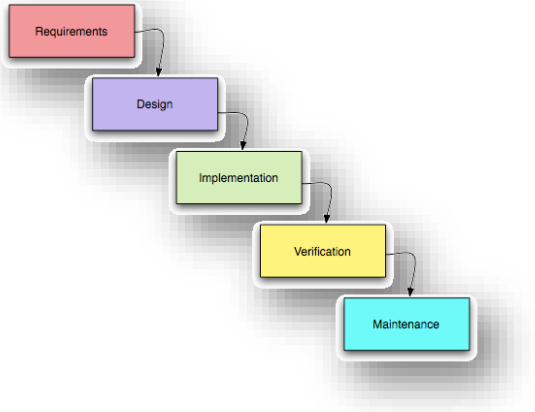
Learning



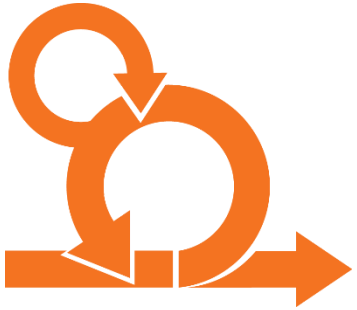

Requirements
Design
Implementation & Unit Testing
Integration & System Testing
Operation & Maintenance

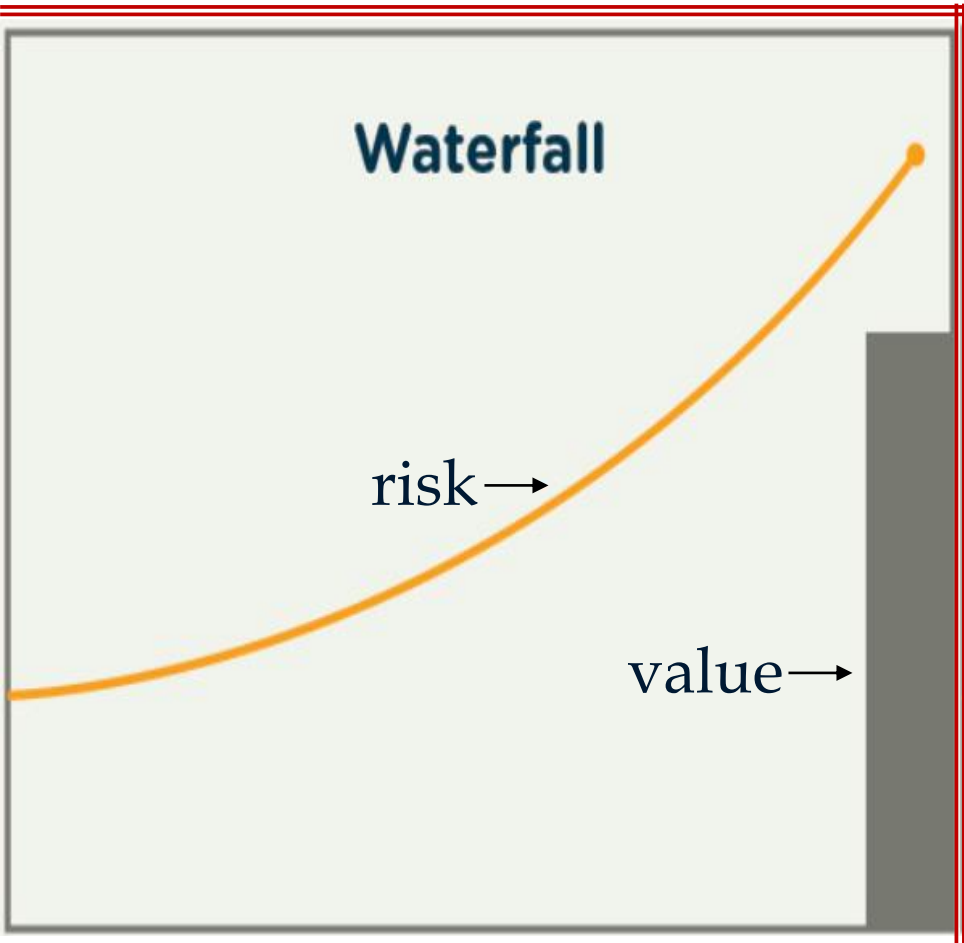

gell facilitated by @caurasa
Learn
Unlearn
Relearn
– Toffler
as cited in
"NOW YOU SEE IT"
by @CathyNDavidson

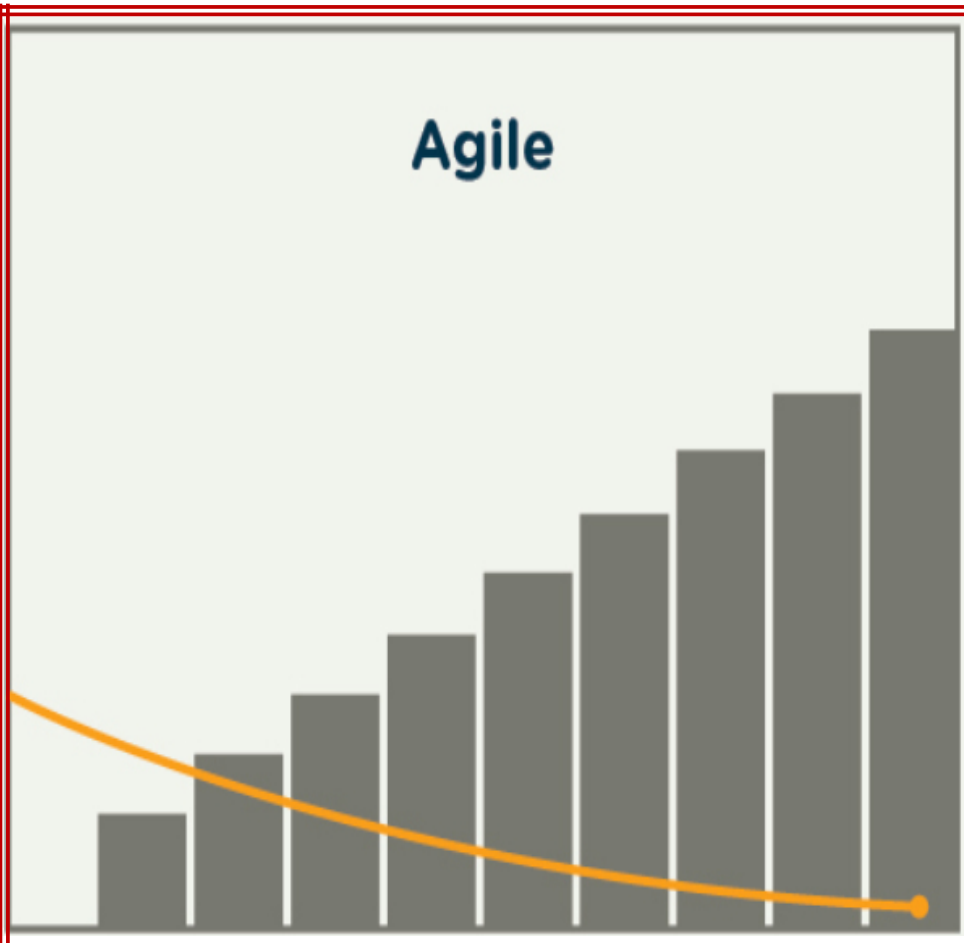
The Backwards Brain Bicycle



Value vs Risk

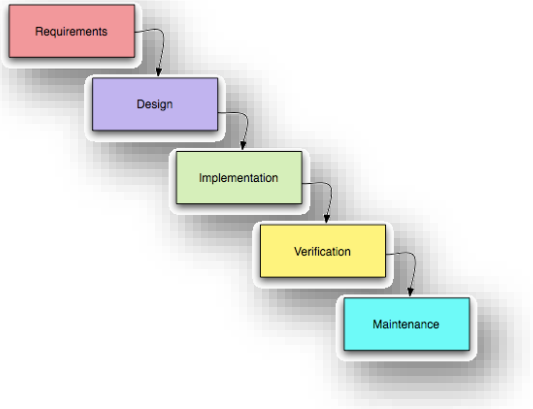




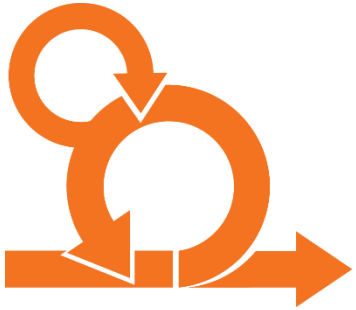


Software is cheapest in lots of SMALL cartons

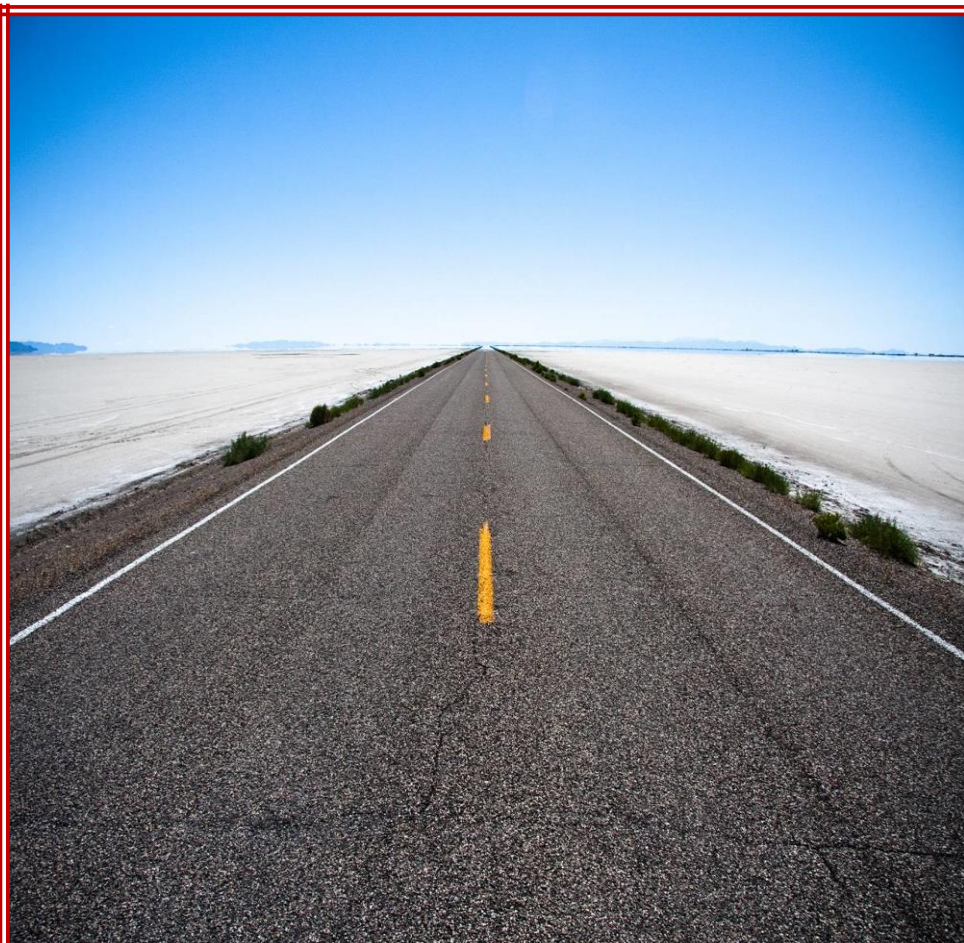
Milk is cheapest in BIG cartons

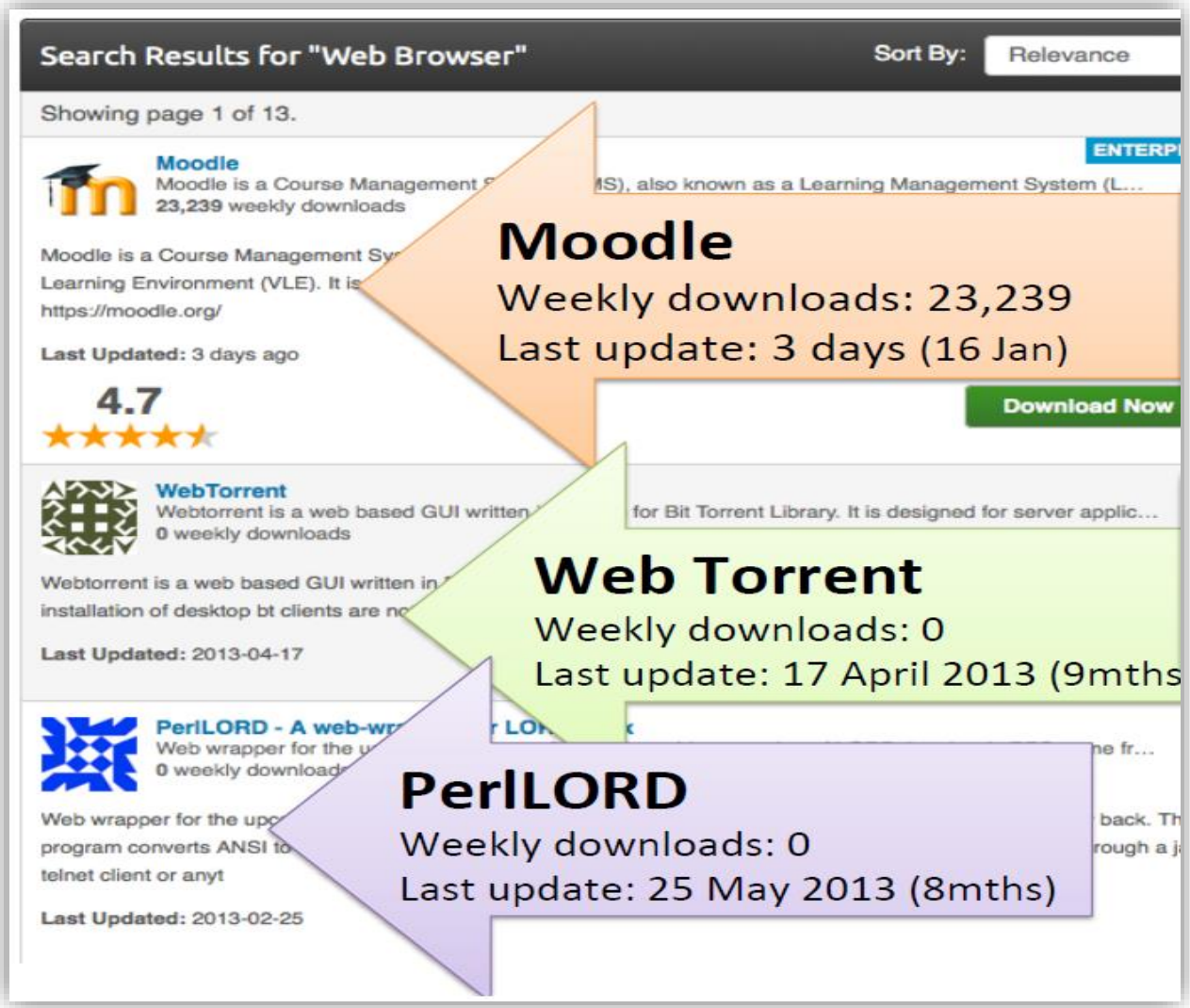

Requirements
Design
Implementation
Verification
Maintenance

Definition

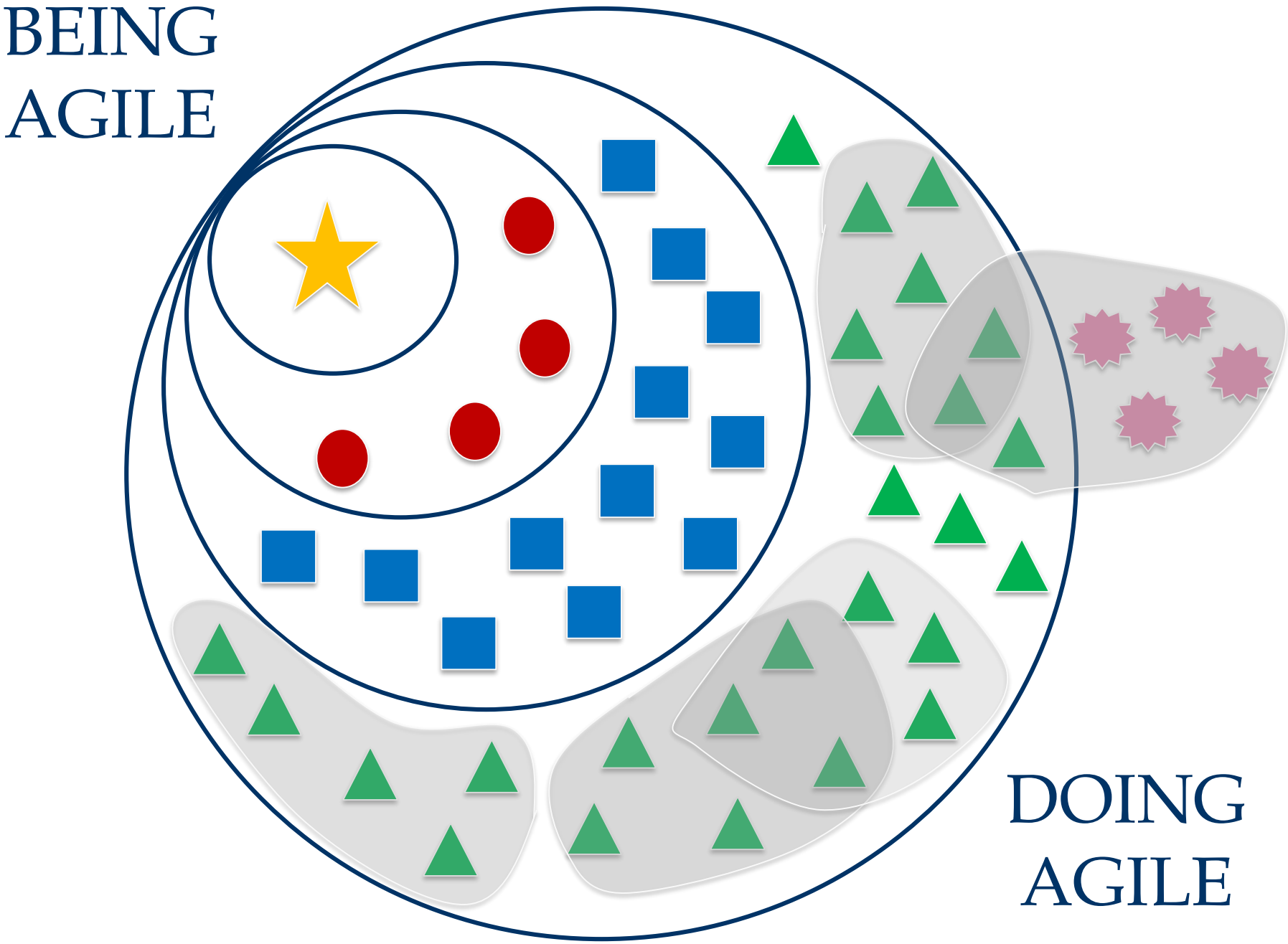


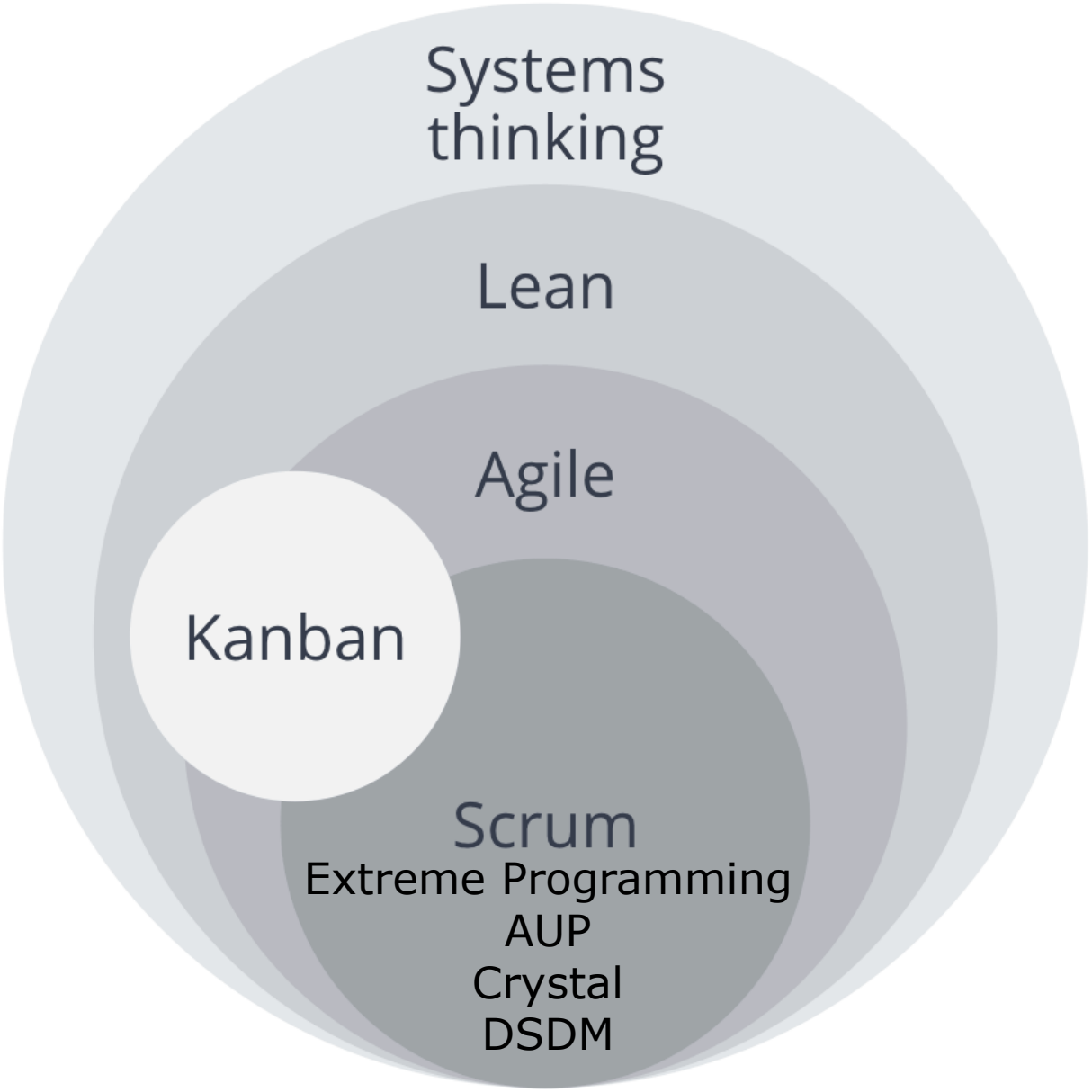
THE END





1. If they use it, it will change
2. Only dead software stops changing


BEING
AGILE
DOING
AGILE


Systems thinking
Lean
Agile
Kanban
Scrum
Extreme Programming
AUP
Crystal
DSDM

doing agile

development
software
change
processes
process
code
principles
team
work
working
project
product
techniques
lifecycle
must
significant
documentation
rate
instead
projects
methods
business
early
waterfall
study
iterative
changing
continuous
functional
productivity
manifesto
stakeholders
specifications
manufacturing
using
agility
integration
detailed
activity
larman
quality
developers
knowing
teams
however
defect
valuable
evidence
difficult
may
people
progress
system
per
give
also
value
common
transition
used
experience
design
often
approach
many
help
hard
adopting
throughout
features
impact
month
serve
resulted
client
plan
goal
better
less
managers
first
frequently
new
defects
communication
various
behavior
adopt
supporting
following
lines
structure
complete
make
feedback
one
writing
develop
guide
clients
weeks
discourage
found
collaboration
systems
requirement
environment
releasing
always
phase
claim
specification
activities
respond
little
conceptual
tools
come
source
amount

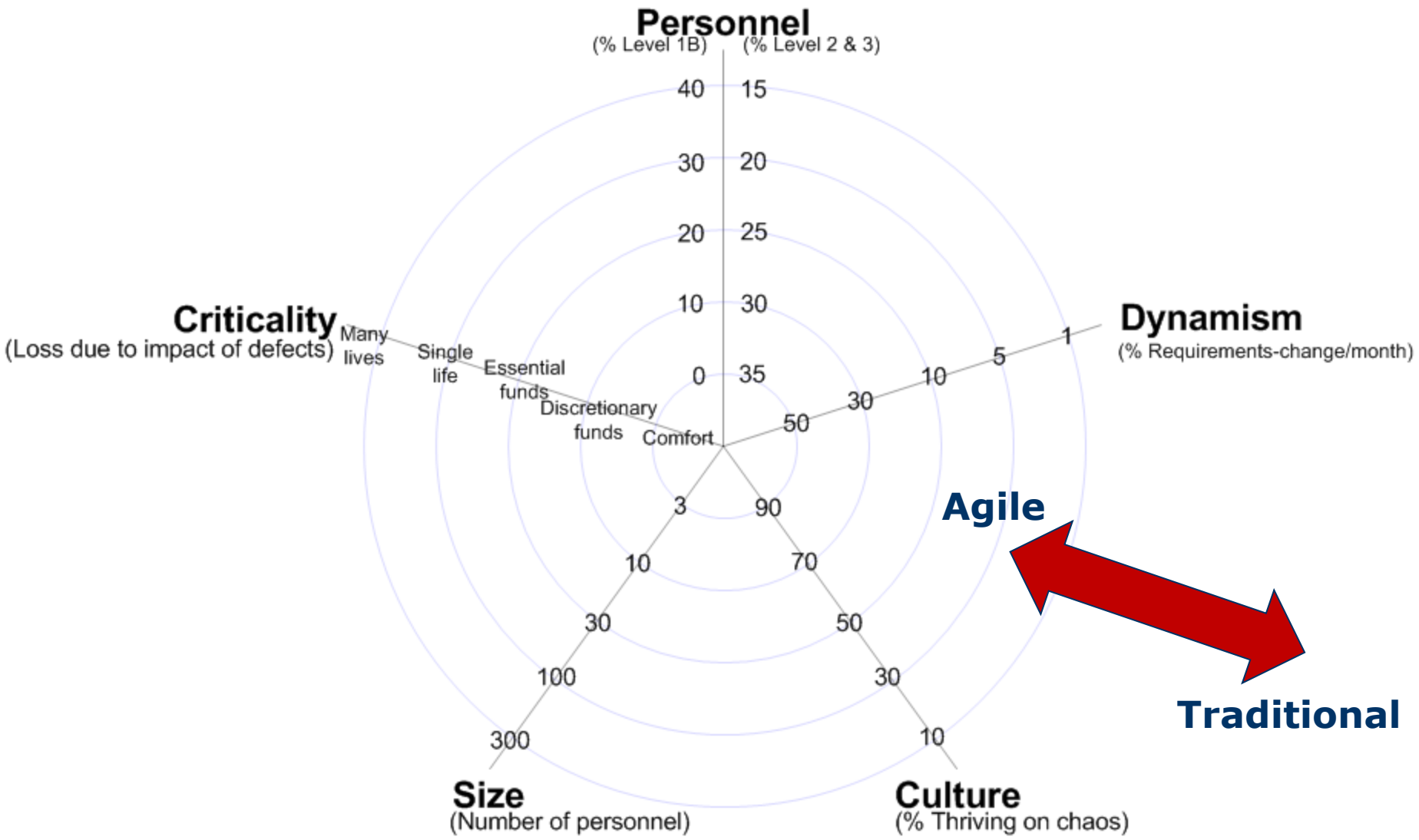
Levels of Software Method Understanding and Use

Level	Characteristics
3	Able to revise a method (break its rules) to fit an unprecedented new situation
2	Able to tailor a method to fit a precedented new situation
1A	With training, able to perform discretionary method steps (e.g., sizing stories to fit increments, composing patterns, compound refactoring, complex COTS integration). With experience can become Level 2.
1B	With training, able to perform procedural method steps (e.g. coding a simple method, simple refactoring, following coding standards and CM procedures, running tests). With experience can master some Level 1A skills.
-1	May have technical skills, but unable or unwilling to collaborate or follow shared methods.

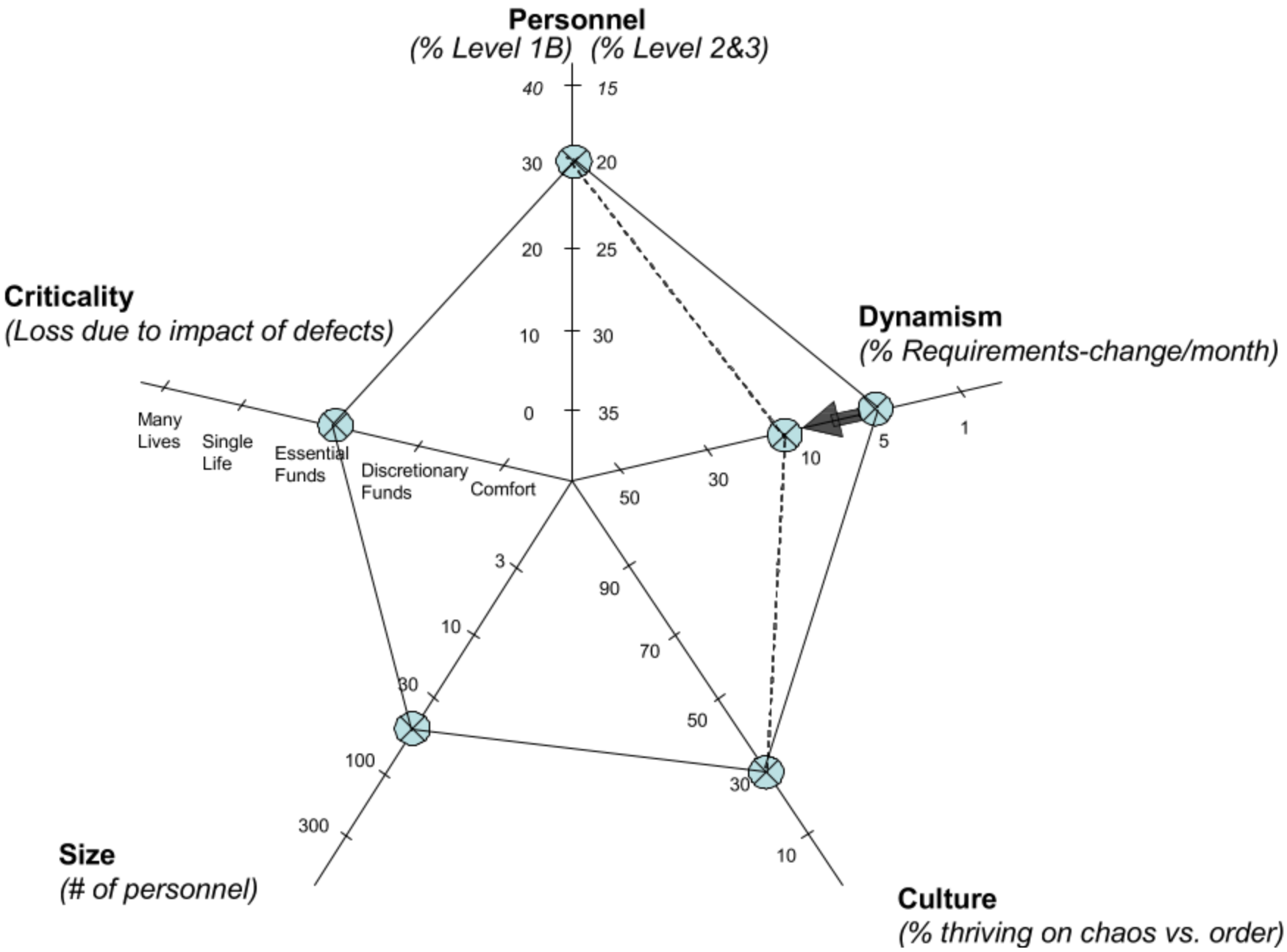
From "Agile Software Development", A. Cockburn (Addison-Wesley, 2002)

Factor	Agility Considerations	Discipline Considerations
Size	Well-matched to small products and teams. Reliance on tacit knowledge limits scalability.	Methods evolved to handle large products and teams. Hard to tailor down to small projects.
Criticality	Untested on safety-critical products. Potential difficultiies with simple design and lack of documentation.	Methods evolved to handle highly critical products. Hard to tailor down to low-criticality products.
Dynamism	Simple design and continuous refactoring are excellent for highly dynamic environments, but a source of potentially expensive rework for highly stable environments.	Detailed plans and Big Design Up Front excellent for highly stable environment, but a source of expensive rework for highly dynamic environments.
Personnel	Requires continuous presence of a critical mass of scarce Cockburn Level 2 or 3 experts. Risky to use non-agile Level 1B people.	Needs a critical mass of scarce Cockburn Level 2 and 3 experts during project definition, but can work with fewer later in the project—unless the environment is highly dynamic. Can usually accommodate some Level 1B people.
Culture	Thrives in a culture where people feel comfortable and empowered by having many degrees of freedom.	Thrives in a culture where people feel comfortable and empowered by having their roles defined by clear policies and procedures.

from "Balancing Agility and Discipline", Barry Boehm & Richard Turner



from "Balancing Agility and Discipline", Barry Boehm & Richard Turner


Personnel
(% Level 1B) (% Level 2&3)
40
30
20
10
0
15
20
25
30
35
Criticality
(Loss due to impact of defects)
Many Lives
Single Life
Essential Funds
Discretionary Funds
Comfort
Dynamism
(% Requirements-change/month)
1
5
10
30
50
Size
(# of personnel)
300
100
30
10
3
Culture
(% thriving on chaos vs. order)
10
30
50
70
90

Christopher Avery

"We can never direct a living system, only disturb it and wait to see the response…

We can't know all the forces shaping an organization we wish to change, so all we can do is provoke the system in some way be experimenting with a force we think might have some impact, then watch to see what happens."