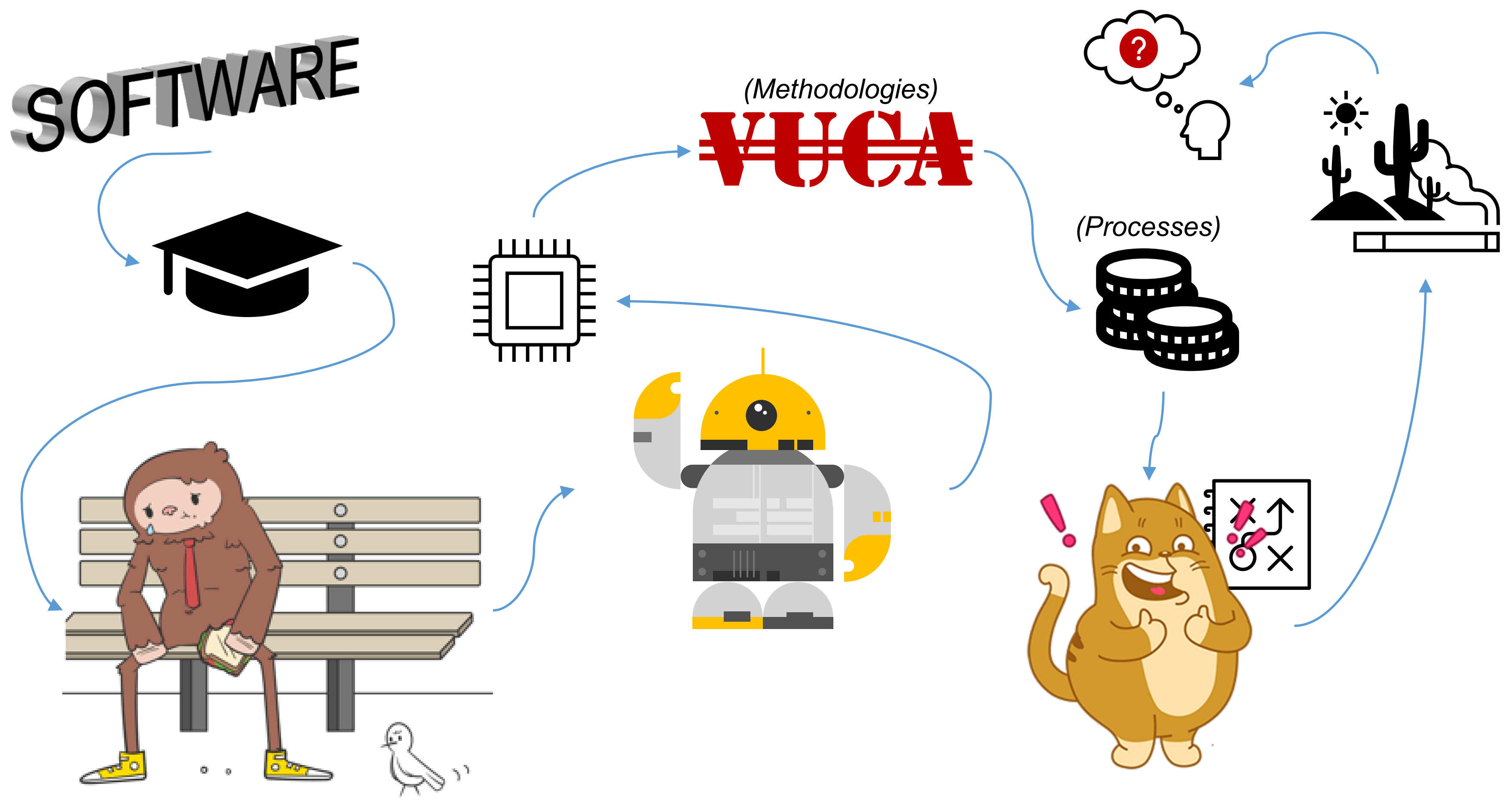
SMU SINGAPORE MANAGEMENT UNIVERSITY

School of Computing and Information Systems

Software Project Management – IS212

Software Development Process

Where Did We Leave Off…



Software Development Processes

Objectives

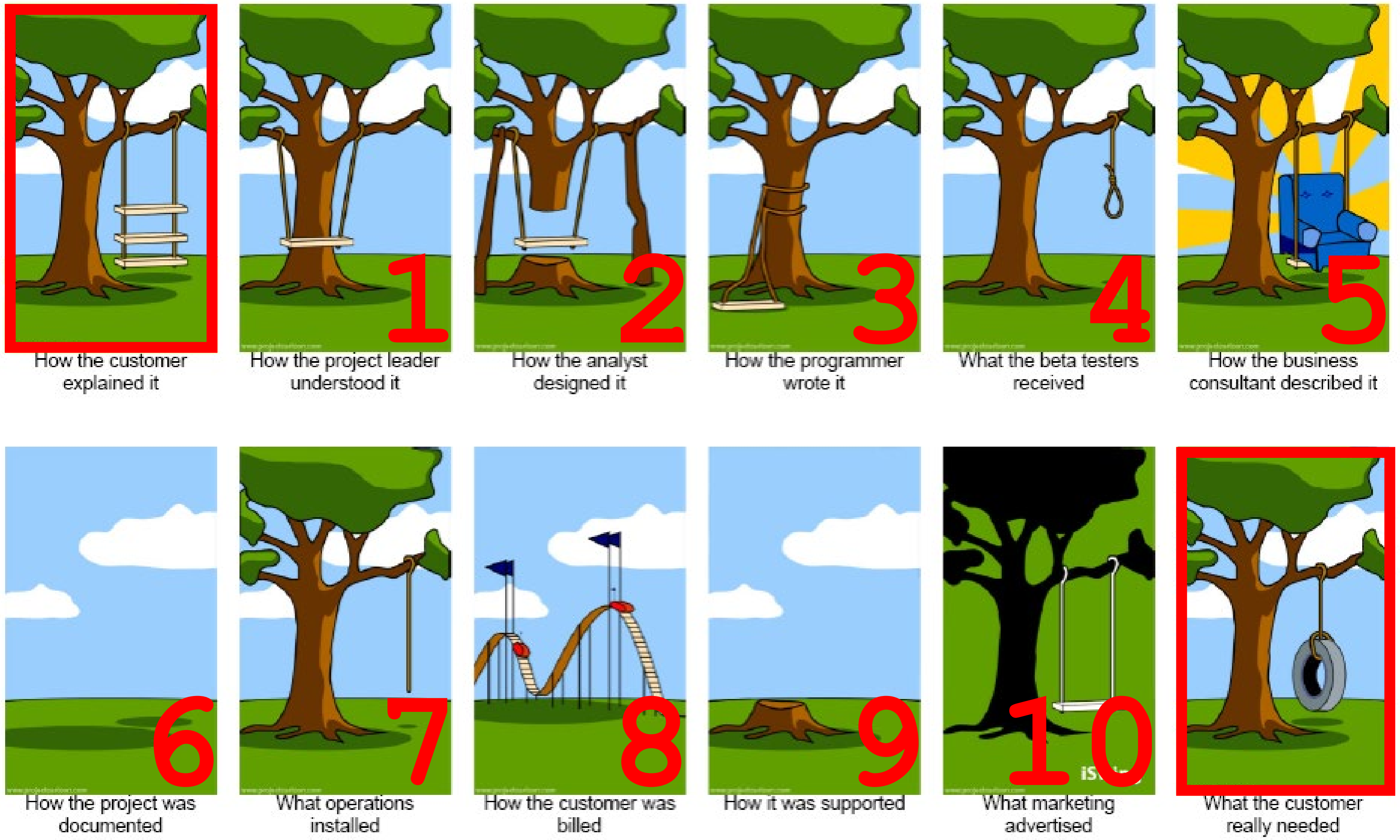
On completing this module, you will be able to:

- Summarise the key traditional and agile software development methods
- Identify the pros and cons of each

Topics

- Traditional software development methods: waterfall, RUP, spiral, prototyping
- Agile methods: extreme programming (XP), scrum, Kanban *(lean)*

A classic comic…


How the customer explained it
1
How the project leader understood it
2
How the analyst designed it
3
How the programmer wrote it
4
What the beta testers received
5
How the business consultant described it
6
How the project was documented
7
What operations installed
8
How the customer was billed
9
How it was supported
10
What marketing advertised
What the customer really needed

Software Development Methods

Traditional

- Waterfall
- RUP
- Spiral
- Prototyping

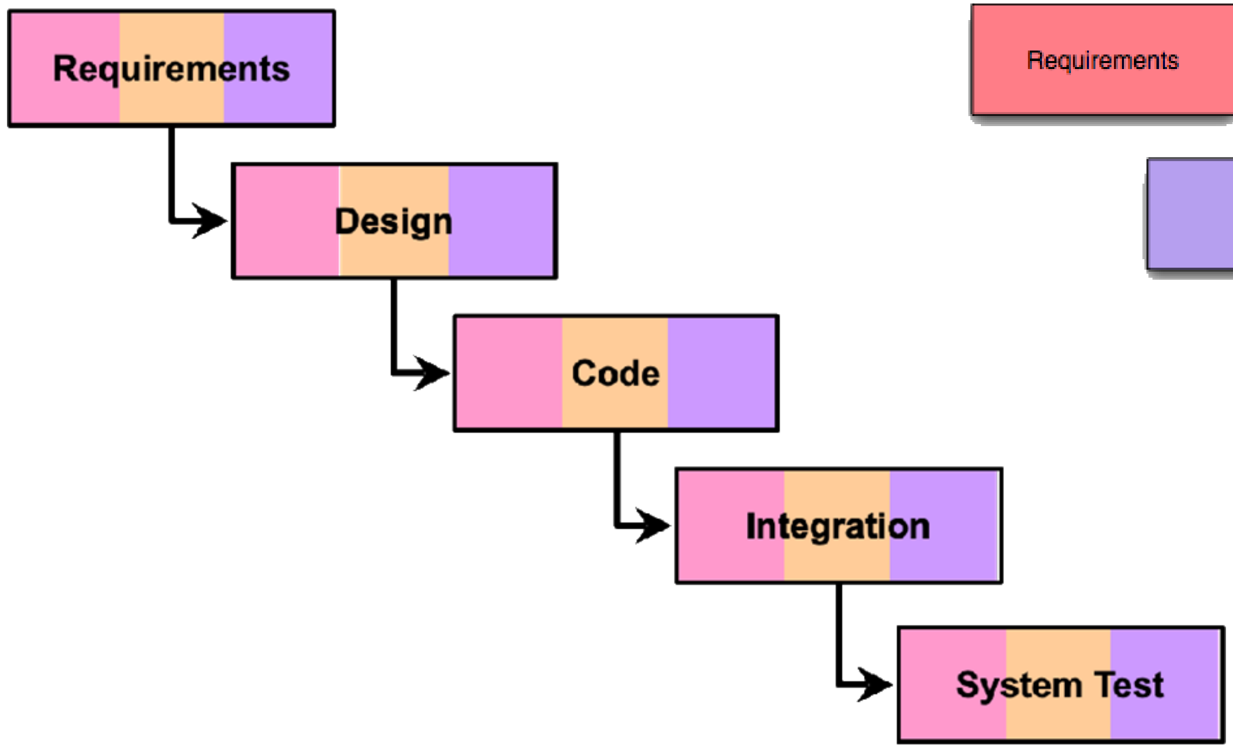
Agile

- Extreme Programming
- Scrum
- Kanban

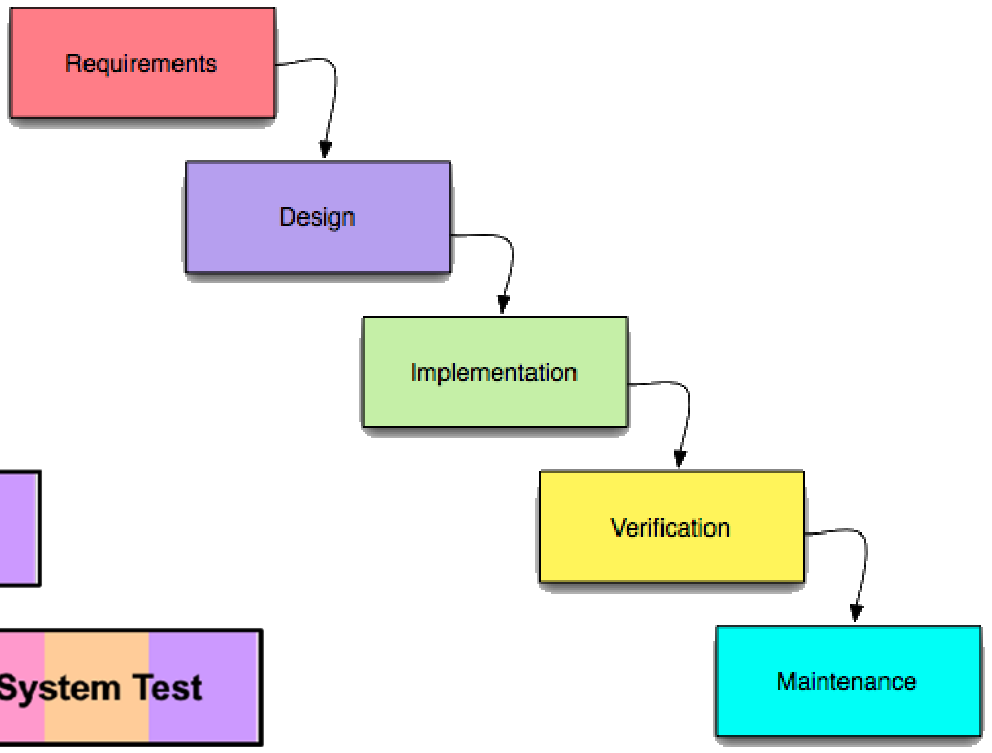
Waterfall

Idea: break software development into linear, sequential phases

(variant)



(original)

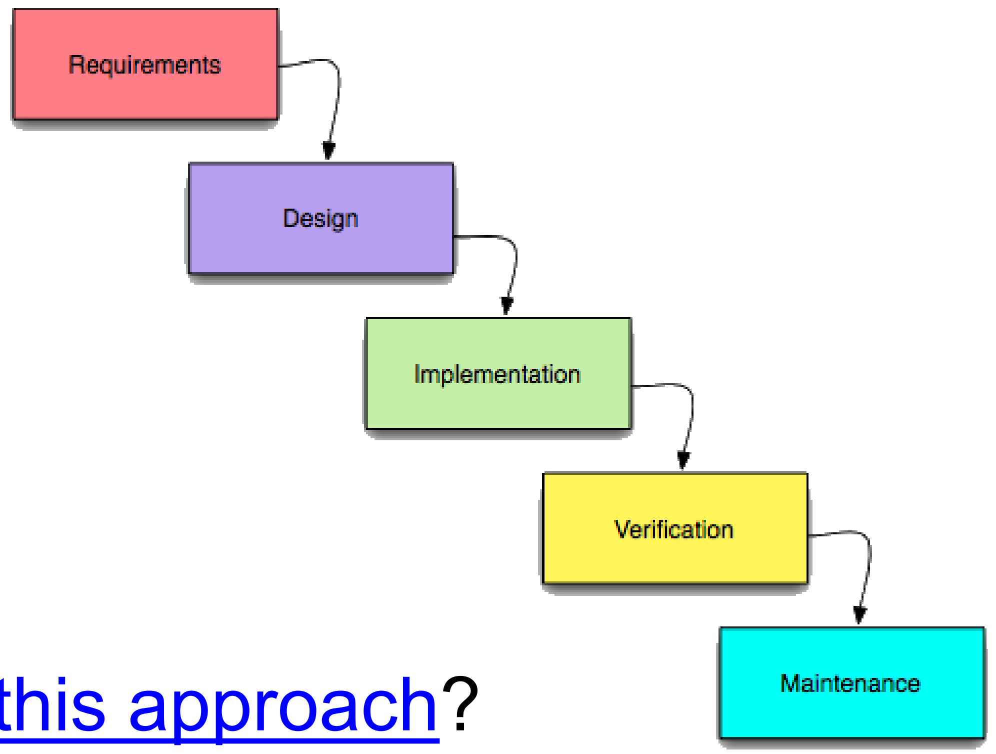


Waterfall – Discussion

Observations

- Partitioning into phases
- Front-loading customer contact
- Back-load testing

(original)



Questions

- What are the pros and cons?
- In what situations would you use this approach?

Waterfall – Discussion

Pros	Cons	When to Use
Simple to understand; easy to use	Unable to cater for ‘non-simple’ changes	Requirement is clear and stable
Each phase has specific deliverables and review process	Integration is a ‘big bang’ activity near the end	Scope is relatively not big and complicated
Easy to manage – clearly defined stages; well understood milestones	‘Working’ software only at the end	
Process and results well documented		

Waterfall – Discussion

1. COMPLETE PROGRAM DESIGN BEFORE ANALYSIS AND CODING BEGINS
2. DOCUMENTATION MUST BE CURRENT AND COMPLETE
3. **DO THE JOB TWICE IF POSSIBLE**
4. TESTING MUST BE PLANNED, CONTROLLED AND MONITORED
5. **INVOLVE THE CUSTOMER**

Real-world software projects are iterative

Software Development Methods

Traditional

- Waterfall
- RUP
- Spiral
- Prototyping

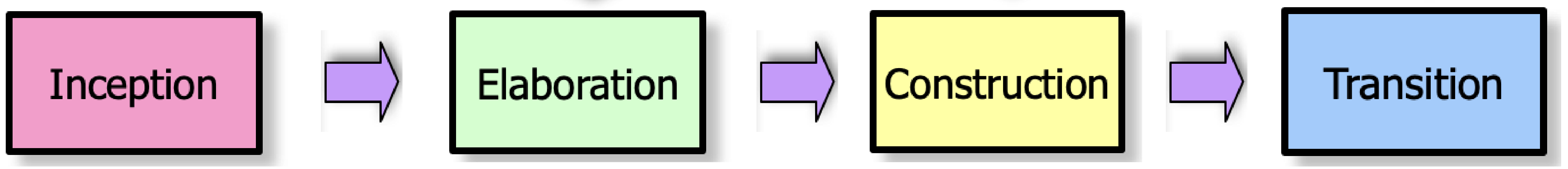
iterative traditional methods —>

Agile

- Extreme Programming
- Scrum
- Kanban

Rational Unified Process (RUP)

- The Rational Unified Process (RUP) consists of four phases
  - i.e. key 'milestones' for the project, similar to waterfall



understand ***what*** product to build

understand ***how*** to build the product

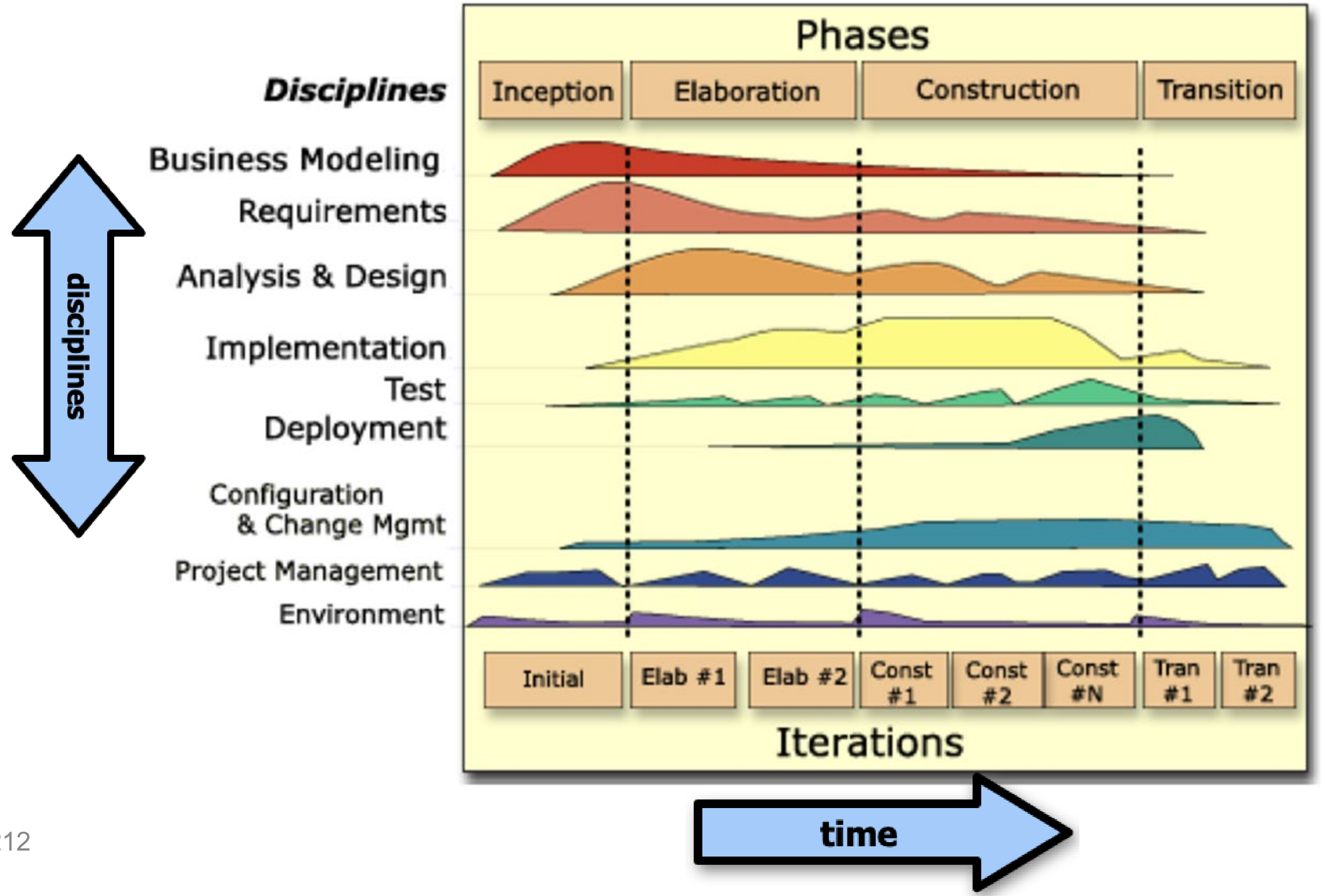
actually ***build*** the product

validate and ***deploy*** the product

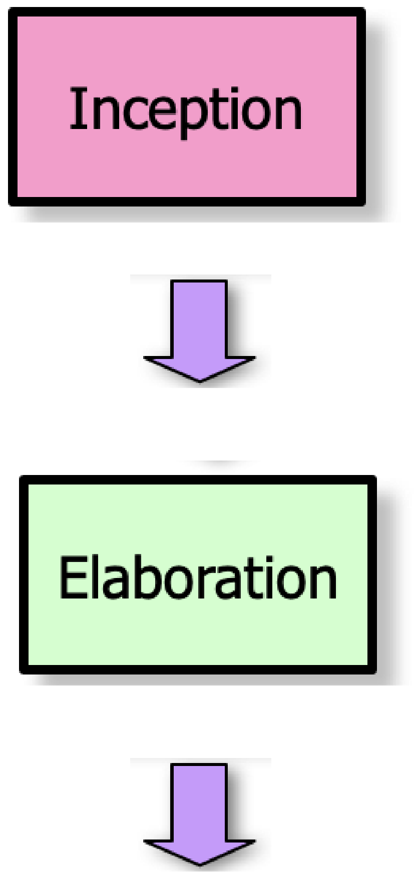
- Various 'engineering disciplines' then cut across all four phases
  - *e.g. business modelling, requirements, implementation, testing*

Rational Unified Process (RUP)

- Unlike waterfall – it explicitly encourages iterations within phases



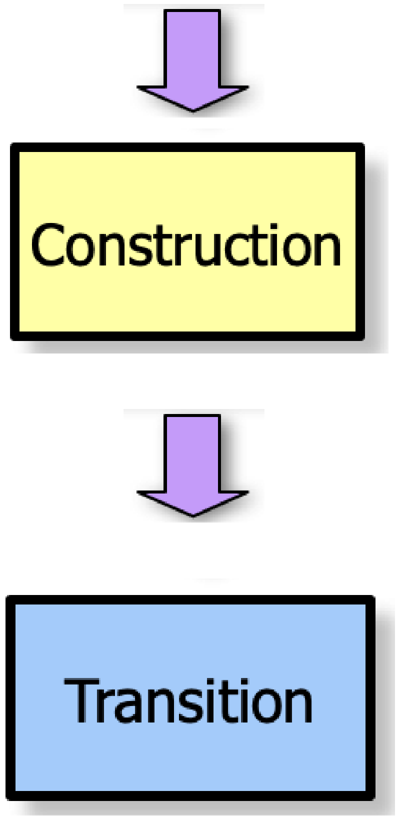
RUP phases – possible outcomes



- Vision document / business case
- Develop high-level requirements / use cases
- Stakeholder concurrence; identify risks

- “80% complete” use case models
- Prototypes for exploring key risks
- Development plan; architecture description

RUP phases – possible outcomes



- Demonstrable prototypes
- Working software components
- Bulk of coding

- Training end users / maintainers
- Evaluate against ‘Inception’ requirements
- ‘Post-mortem’ project analysis

RUP – Discussion

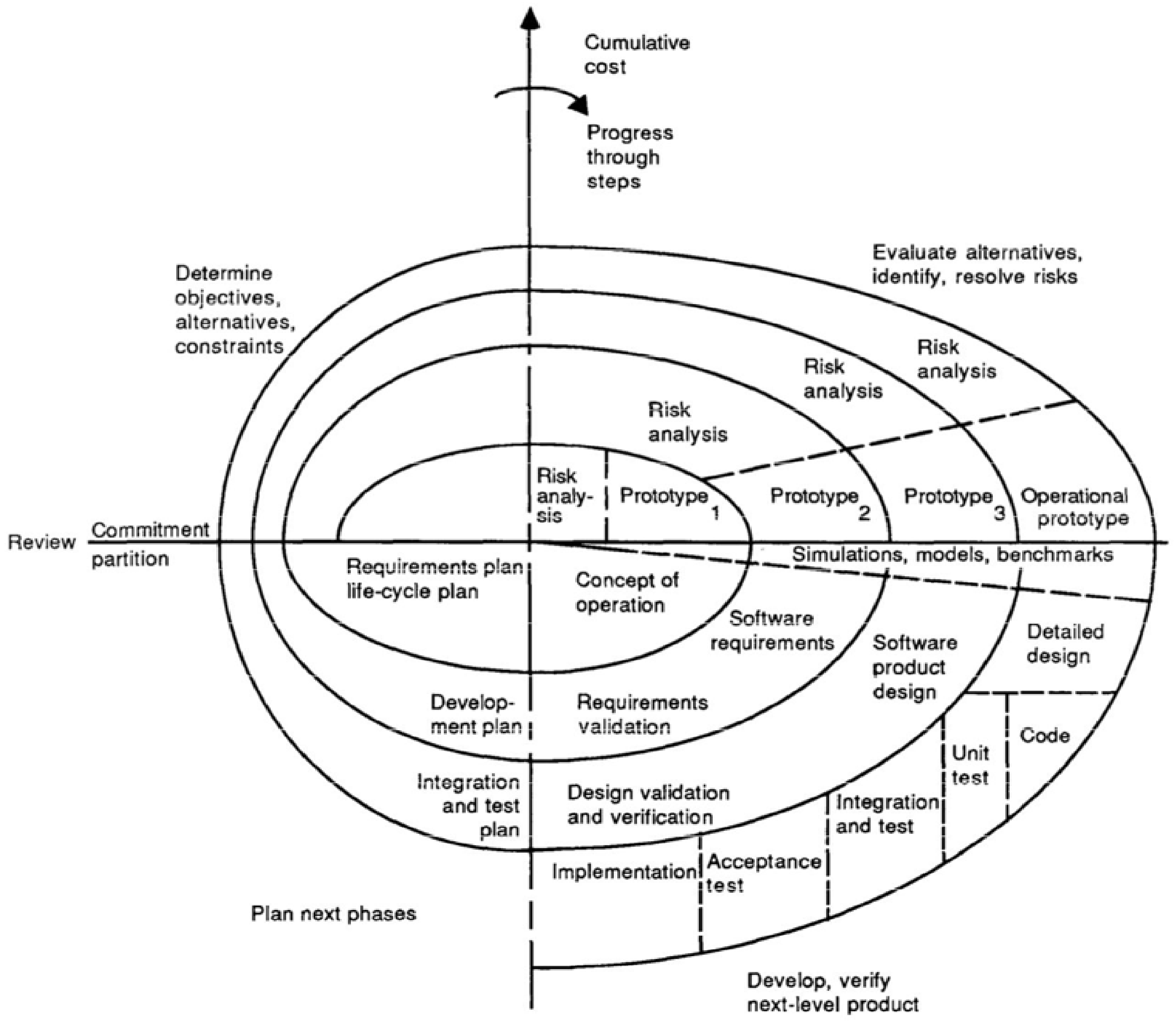
- Support for iteration a big advantage over classic waterfall
- Forces integration to happen throughout the software development
- Testing can happen as early as 'inception' – not left til the end!
- RUP is quite complex; a lot of process 'overhead'
  - Less flexible than modern agile / scrum methods *(see later!)*

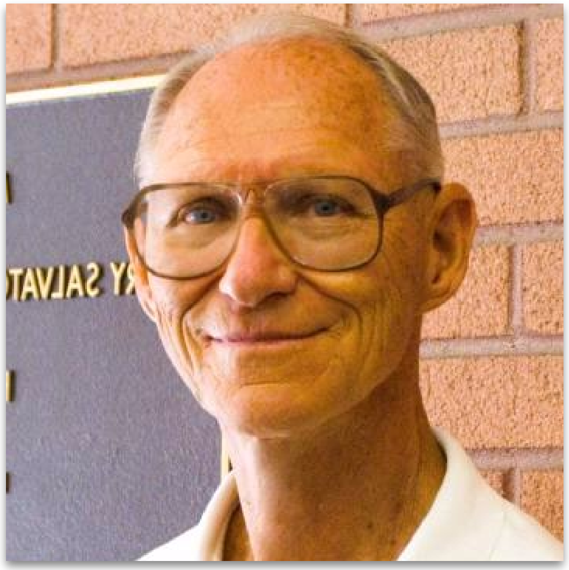
RUP – Discussion

Pros	Cons	When to Use
Allows for changing requirements through iterative cycles of development disciplines	Relies on experienced and proficient team members to plan project	Higher technical complexity, e.g., real-time, distributed, fault tolerant, etc.
Emphasizes the need for necessary and accurate documentation	More documentation as each iteration of development discipline includes fait amount of documentation	Higher managerial complexity, e.g., large scale, contractual, multiple stakeholders, etc.
Continuous rather than back-loading testing	Needs to be customised for effective use	

Spiral

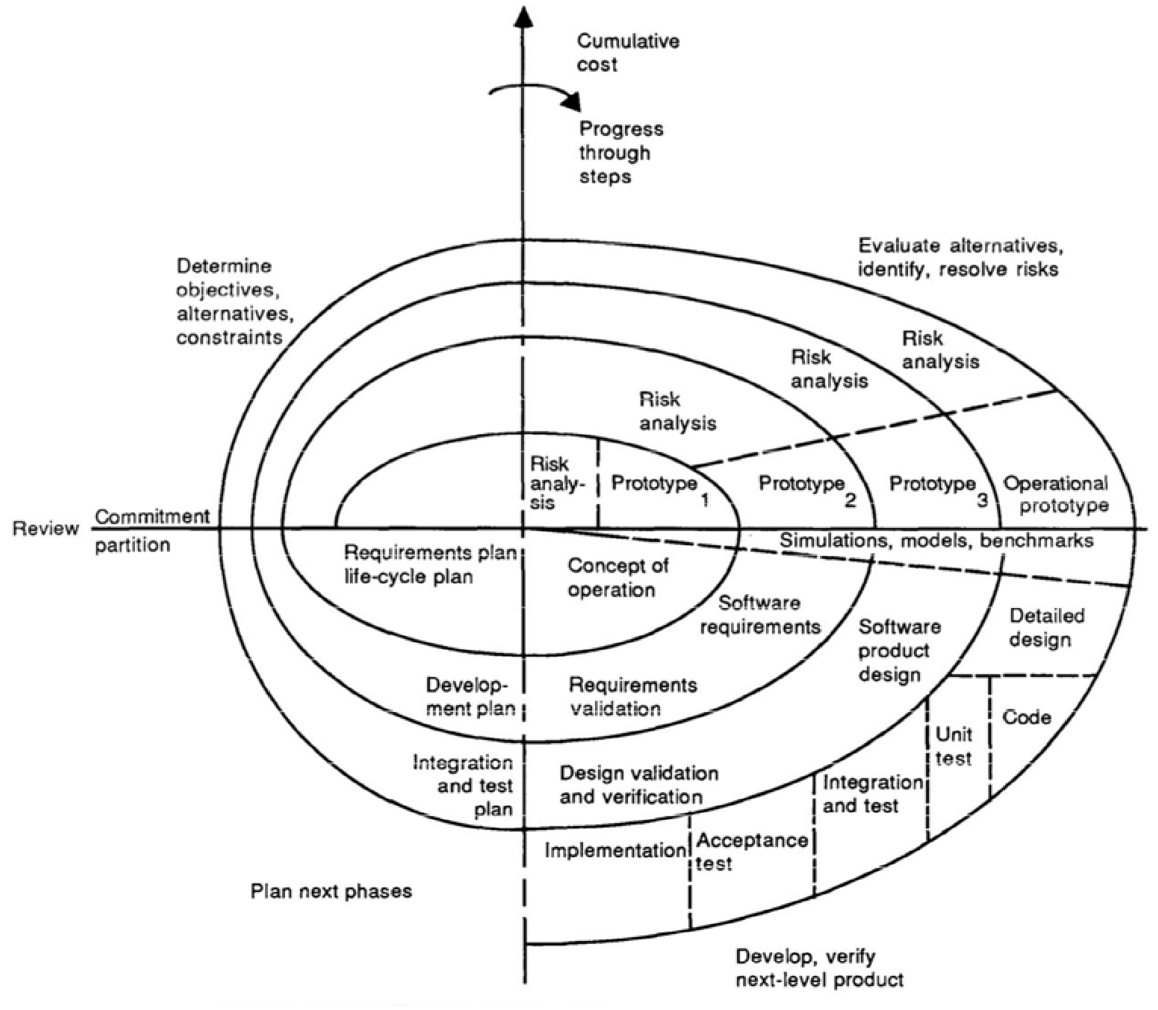
- Spiral – a classic risk-driven method proposed by Barry Boehm





Spiral

- Another iterative process for large systems
- Each loop represents a phase
- No fixed phases – loops flexibly determined according to needs
- Emphasises exploration of alternative options and risks



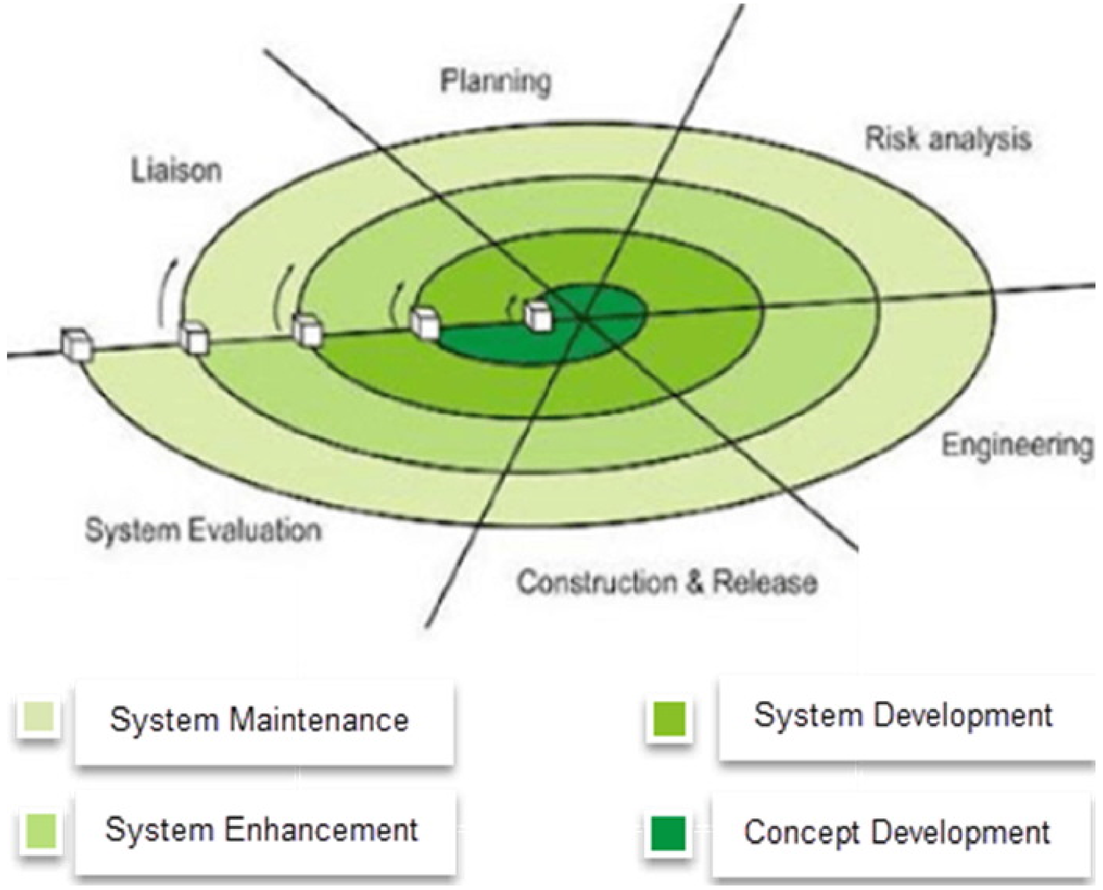
Spiral – Discussion

Observations

- Partitioning into phases
- Risk-driven

Questions

- What are the pros and cons?
- In what situations would you use this approach?

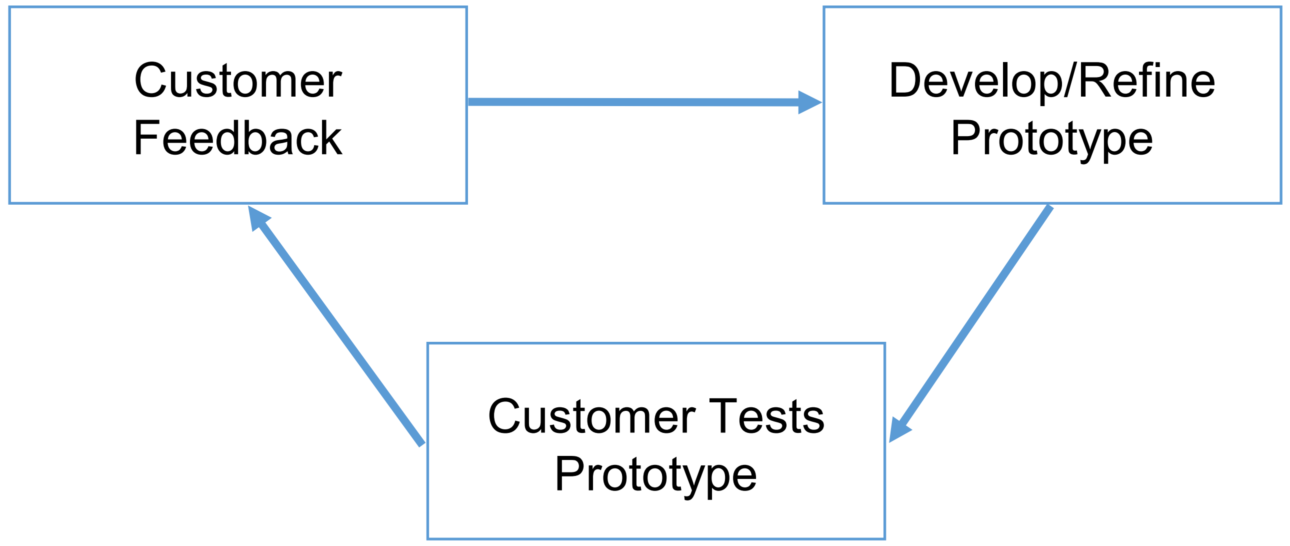


Spiral – Discussion

Pros	Cons	When to Use
Changes can be folded in, and addressed in subsequent phases	Number of phases uncertain and final time/cost may be uncertain	Risky projects
Iterative prototyping helps manage risks	More documentation as each phase includes documentation	Requirements may change at any time
Cost estimation easier as it is for each phase	Final cost estimation harder as uncertain number of phases	Long-term commitment to project is uncertain

Prototyping

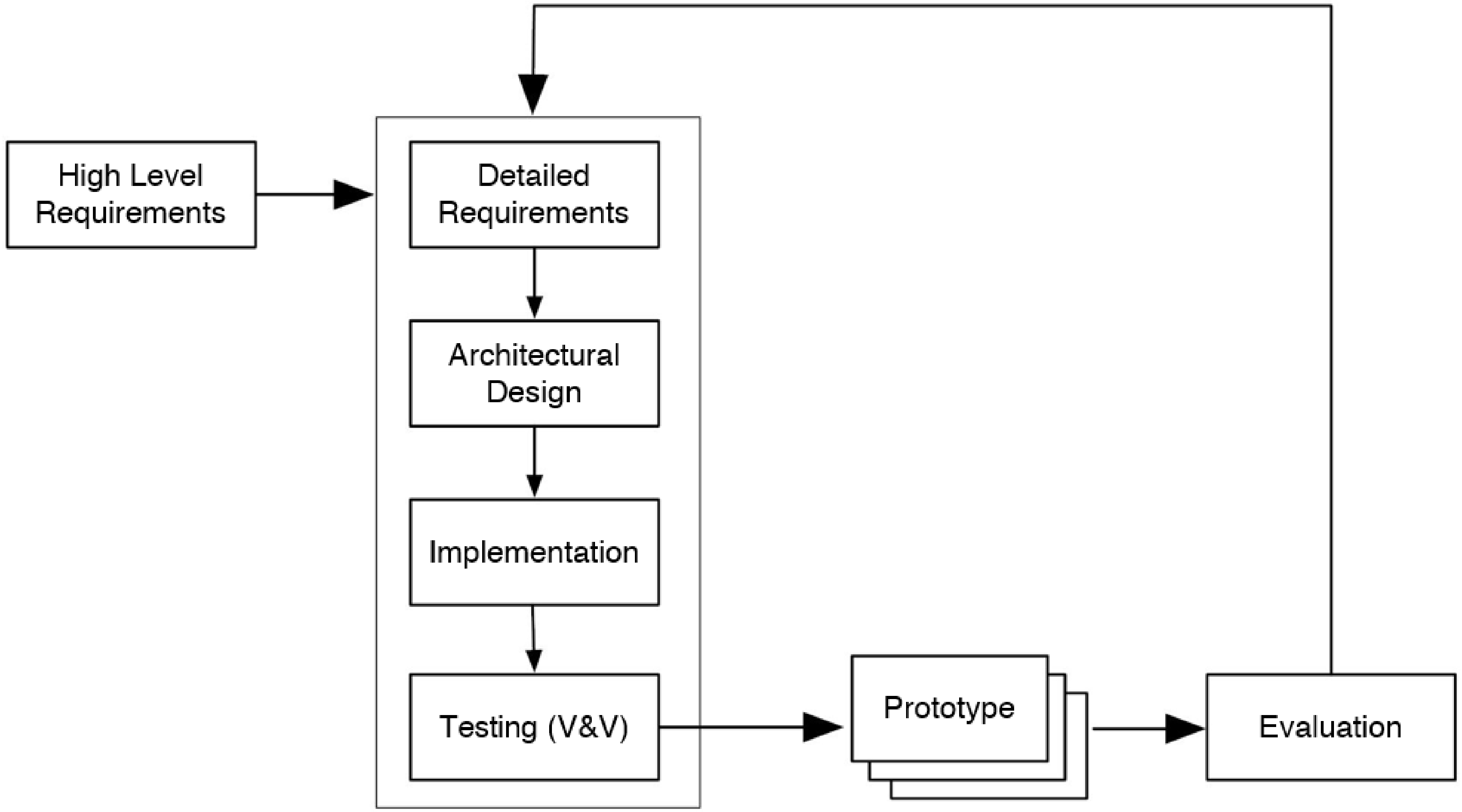
- This final method focuses on the activity of prototyping itself
- Useful for when customers do not know their exact requirements
  - Roughly:



Prototypes can be:
- horizontal (entire application)
- vertical (specific function)

and/or:
- throw-away (just a demo)
- evolutionary (iteratively evolves to become the system)

Prototyping



Prototyping – Discussion

Pros	Cons	When to Use
Customer actively involved in what is being built	Lack of process visibility	"I'll know it when I see it" situations
Risky components can be developed early to assess feasibility	May encourage excessive changes – re-work, waste, frustration…	
Better for small/medium projects (or specific parts of large projects)	Evolutionary prototypes can lead to poor overall systems	

Software Development Methods

Traditional

- Waterfall
- RUP
- Spiral
- Prototyping

Agile

- Extreme Programming
- Scrum
- Kanban

Agile – Motivation

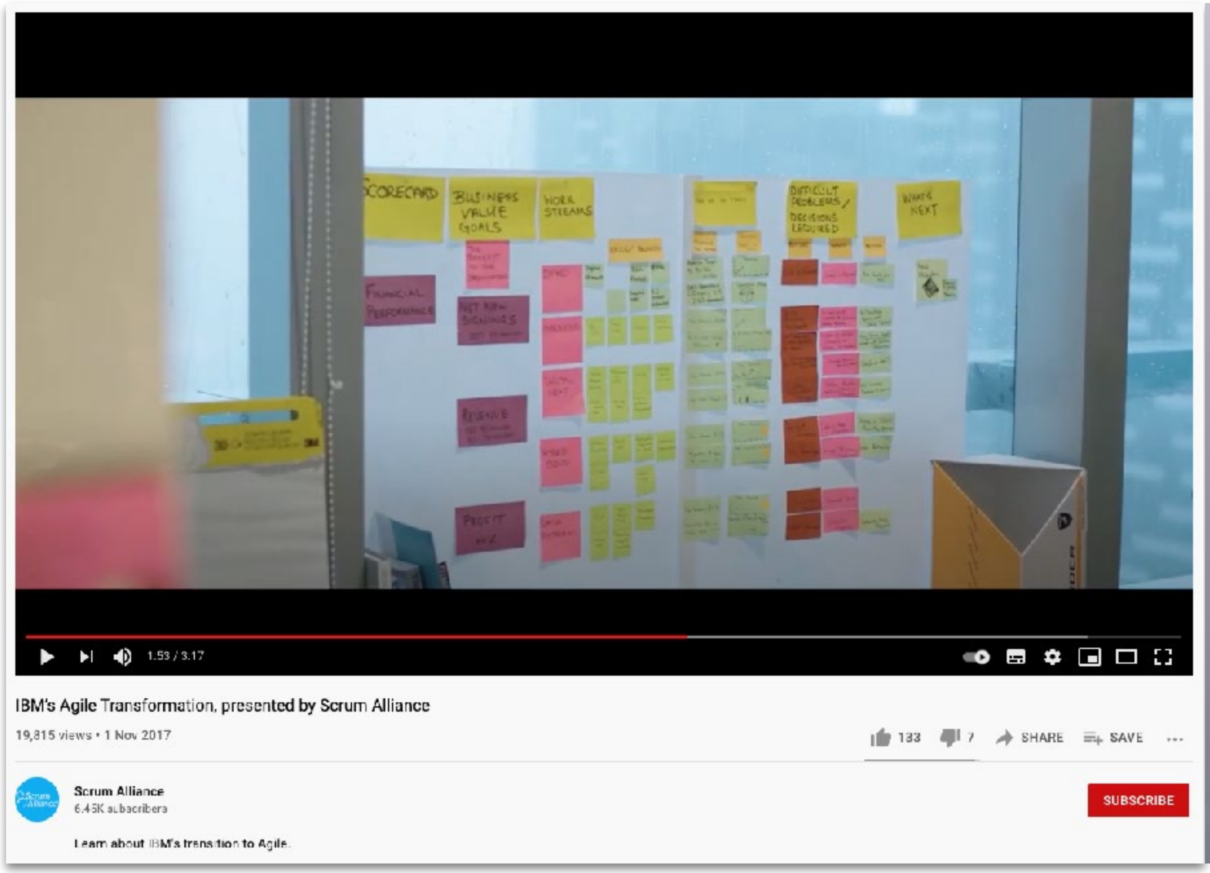
- Traditional heavyweight methods can be overly regulated, planned, and micromanaged, i.e. bureaucratic

- In reaction, a number of lightweight methods started evolving that collectively became known as "agile", emphasising:

Individuals and interactions	**over**	processes and tools
Working software	**over**	comprehensive documentation
Customer collaboration	**over**	contract negotiation
Responding to change	**over**	following a plan

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

Agile – IBM Video


IBM's Agile Transformation, presented by Scrum Alliance

19,815 views • 1 Nov 2017

Scrum Alliance

6.45K subscribers

Learn about IBM's transition to Agile.

IBM – Why Adopt Agile?

https://www.youtube.com/watch?v=Xu0nxyebc6g

Highly iterative and flexible

Working software delivered frequently

Working software (not "WIP") is the measure of progress

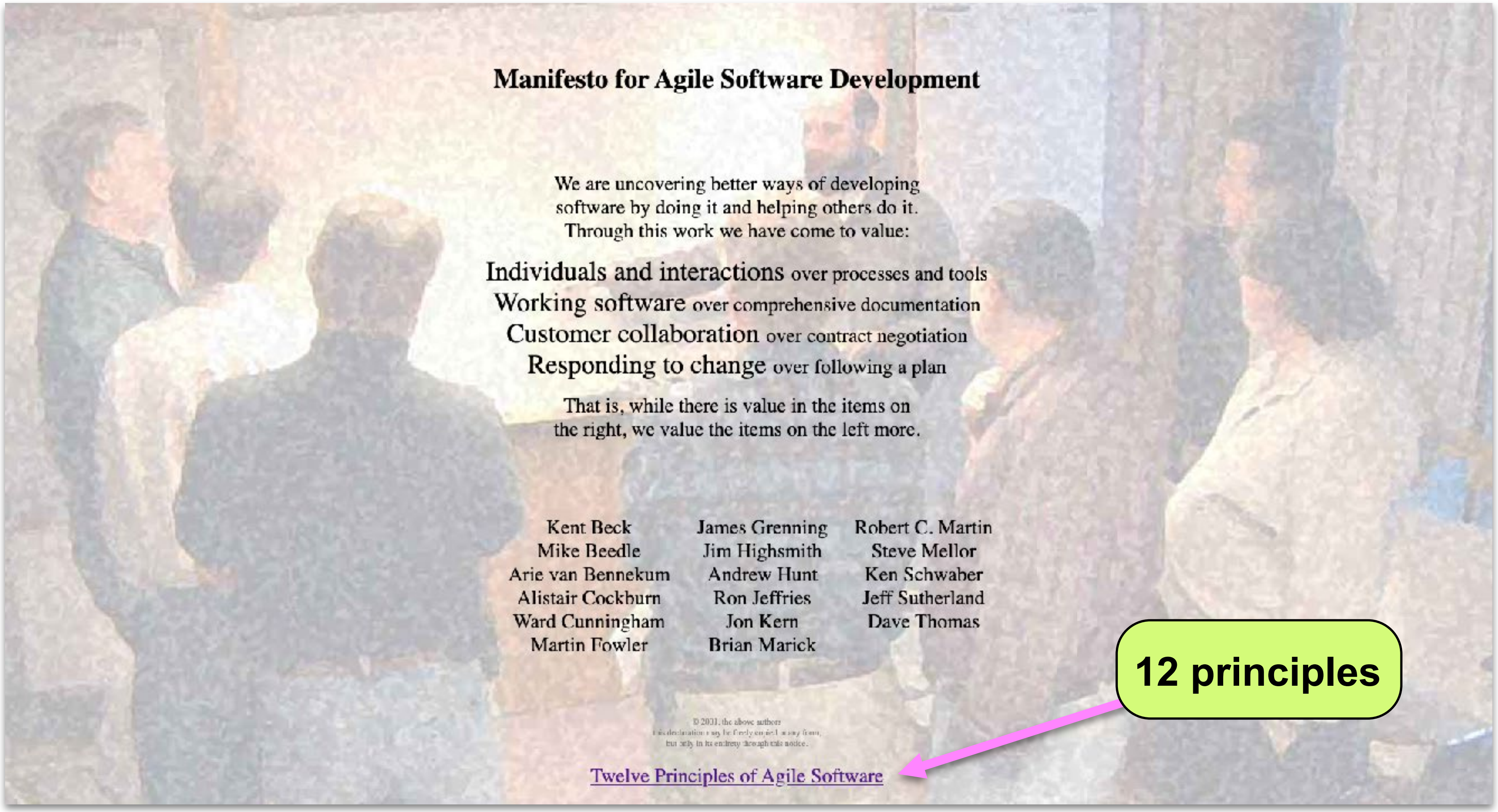
Agile – Key Characteristics

Late changes in requirements – no problem

Close, daily co-operation

Visualise activities and "WIP" to manage team's capacity

Agile Manifesto (2001)



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.

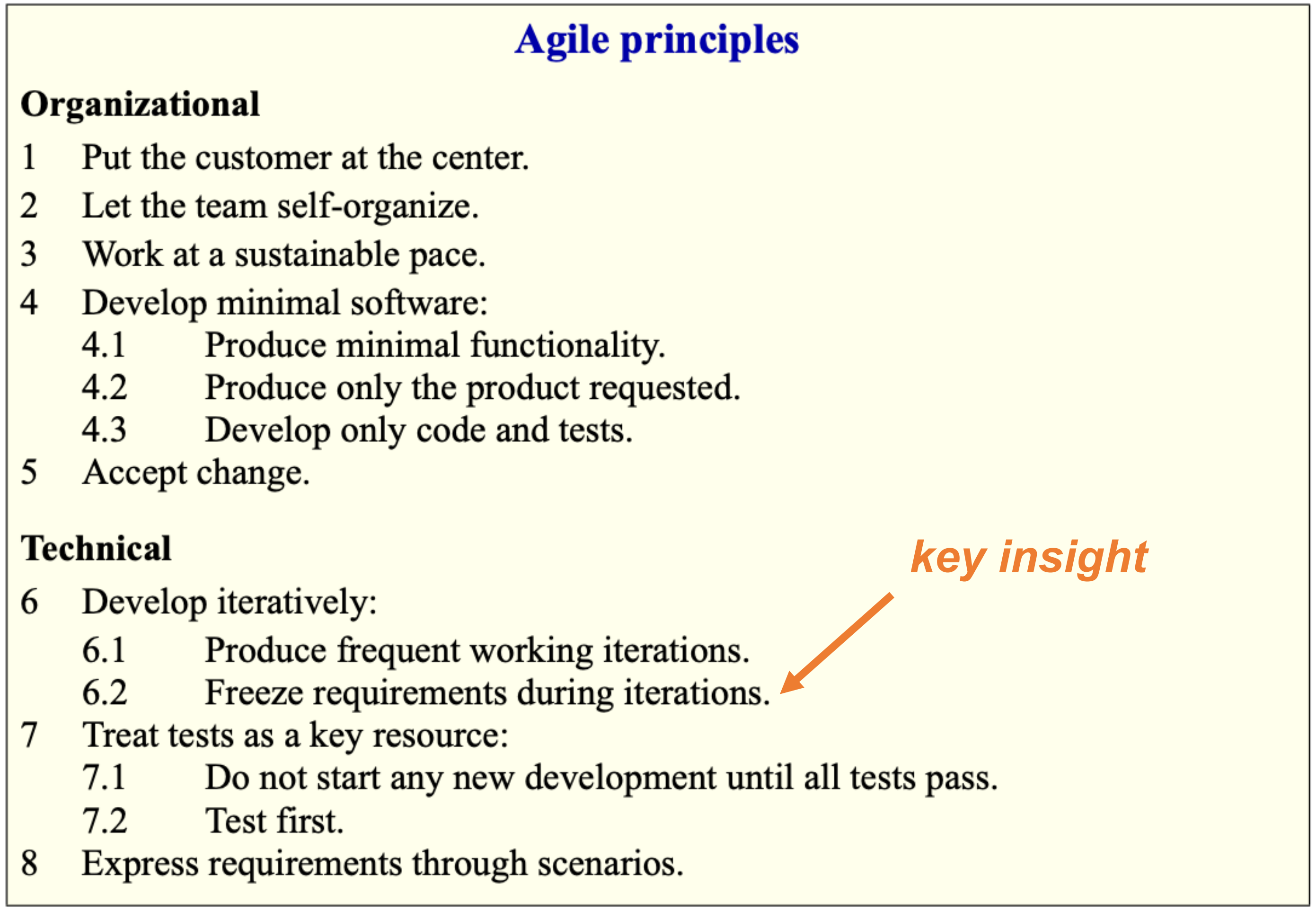
Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

Twelve Principles of Agile Software

12 principles

https://agilemanifesto.org/

Agile Principles *(re-organised; Meyer ’14)*



Agile principles

Organizational

1 Put the customer at the center.
2 Let the team self-organize.
3 Work at a sustainable pace.
4 Develop minimal software:
  4.1 Produce minimal functionality.
  4.2 Produce only the product requested.
  4.3 Develop only code and tests.
5 Accept change.

Technical

6 Develop iteratively:
  6.1 Produce frequent working iterations.
  6.2 Freeze requirements during iterations.
7 Treat tests as a key resource:
  7.1 Do not start any new development until all tests pass.
  7.2 Test first.
8 Express requirements through scenarios.

key insight

Bertrand Meyer

Software Development Methods

Traditional

- Waterfall
- RUP
- Spiral
- Prototyping

Agile

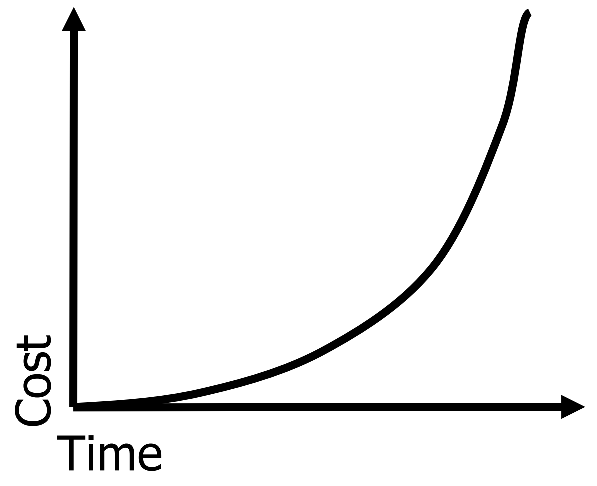
- Extreme Programming
- Scrum
- Kanban

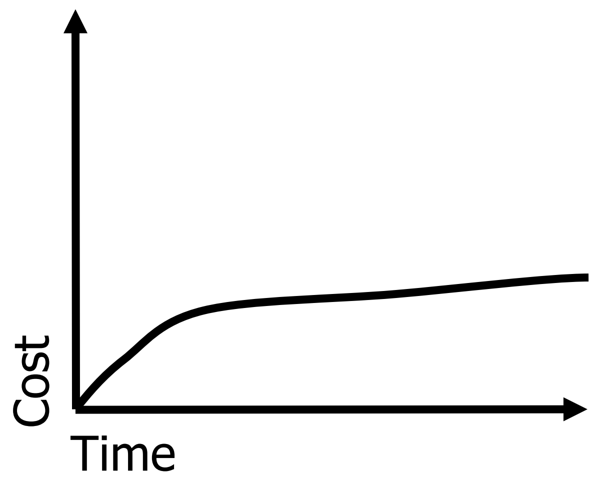
Extreme Programming (XP)

- Premise of extreme programming (XP)

Traditional
- upfront design
- code late
- release when 'done'

XP
- code early
- release early
- continuous design





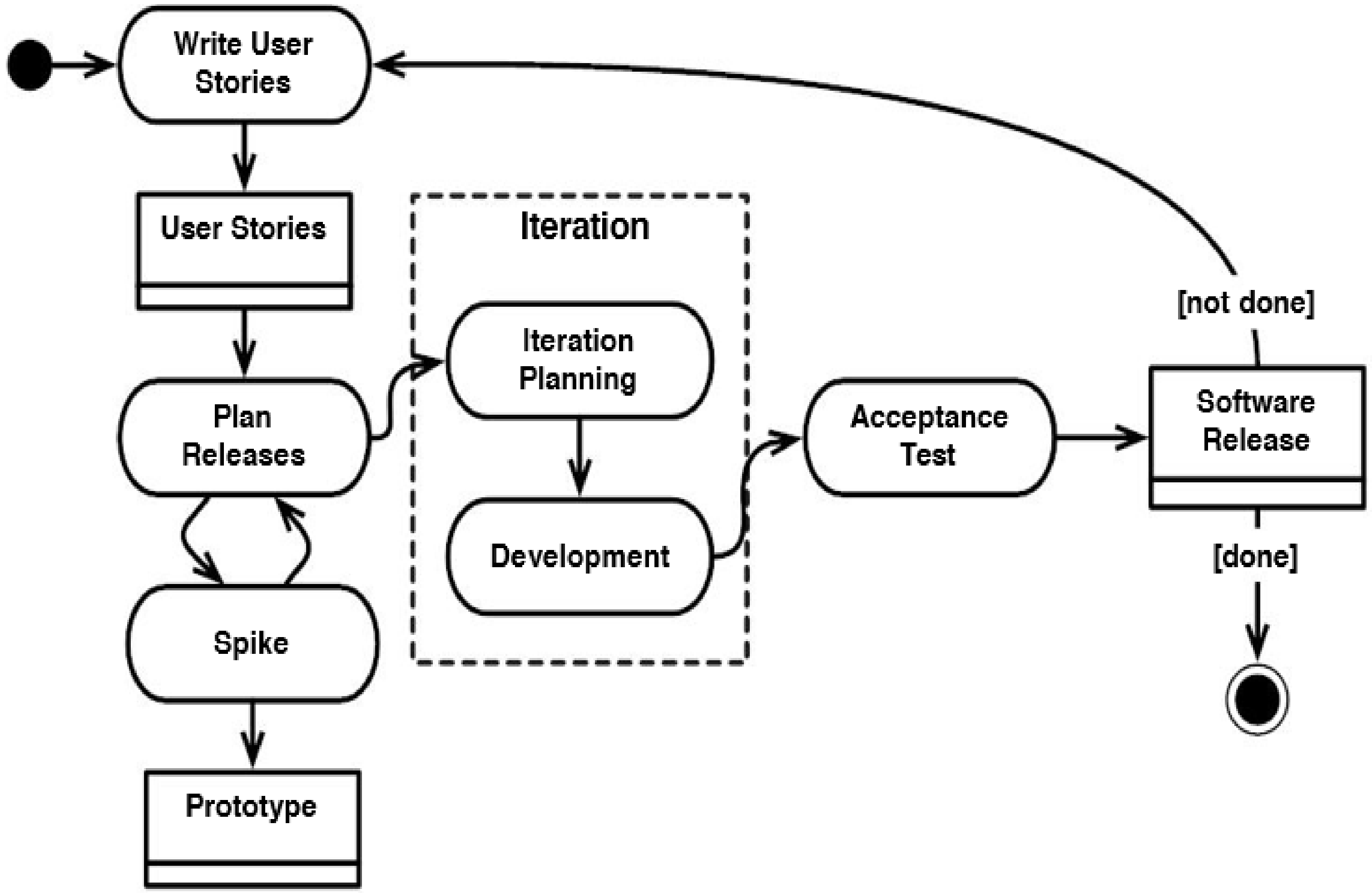
Extreme Programming (XP)

- Focus – customer satisfaction

- XP Principles –

Coding is the core activity	Code doesn't only deliver the solution – use it to explore/explain problems too!
LOTS of testing during dev	Design unit tests or software contracts first – *then* code! (aka **Test-Driven Development**)
Developers and customers communicate directly	Programmer must understand the business requirements to design a technical solution
Regular refactoring	Overall system design is considered during regular **refactoring** exercises

XP Process


Write User Stories
User Stories
Iteration
Iteration Planning
Development
Plan Releases
Spike
Prototype
Acceptance Test
Software Release
[not done]
[done]

12 XP Practices

Fine-scale feedback

- Pair programming
- Planning game
- Test-Driven Development
- Whole team

Shared understanding

- Coding standard
- Collective code ownership
- Simple design
- System metaphor

Continuous process

- Continuous integration
- Design refactoring
- Small releases

Programmer welfare

- Sustainable pace

XP – Impact

- XP provided a 'jolt' that brought attention to agile methods.
- Some see XP as 'dogmatic'; others see it as a consistent, strong view of how programming should be practiced.
- Many of the individual practices promoted by XP have left their mark on industry, especially that:
  - Projects should integrate code all the time;
  - Tests are a key resource, and should be run against code often.

Scrum

"scrum"

- borrowed from rugby (formation of players)

- term introduced to emphasise teamwork

Scrum

What is Scrum?

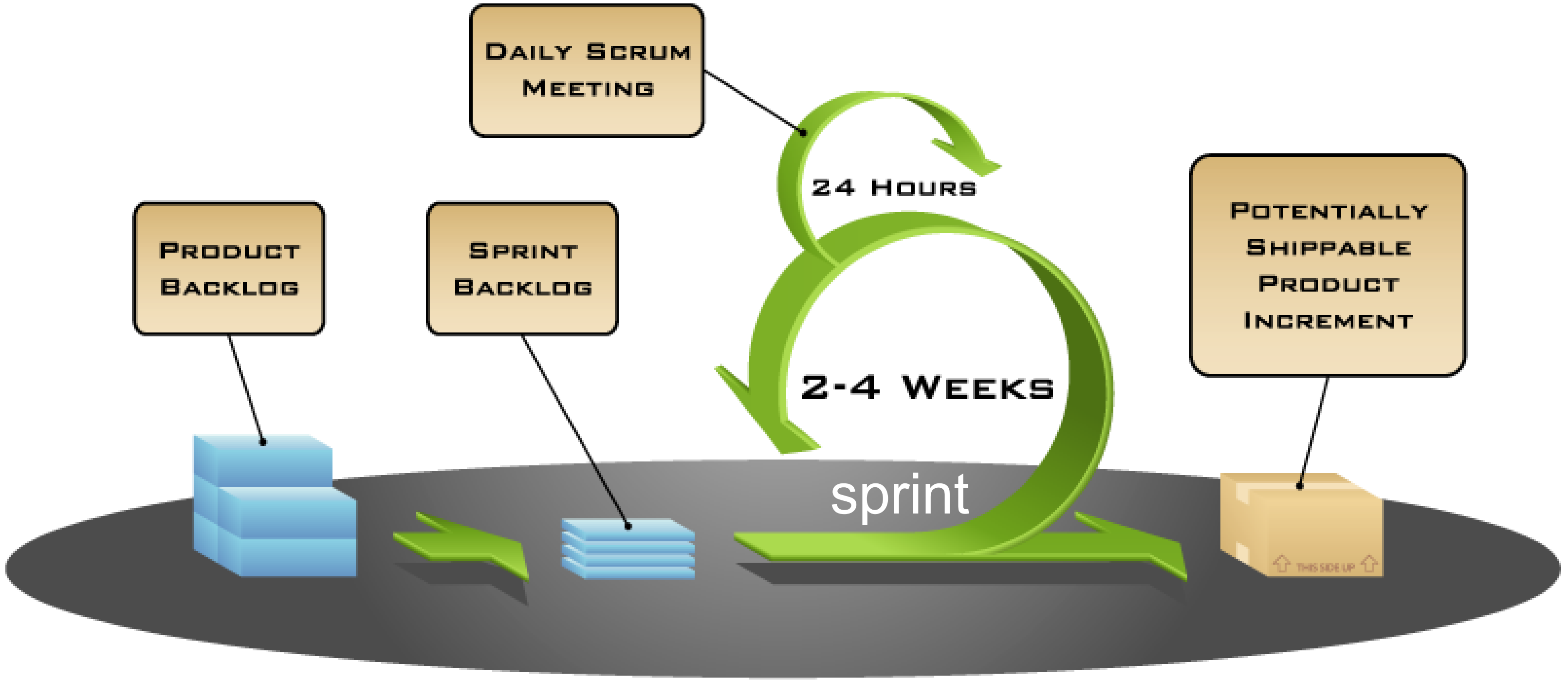
279,742 views • 26 Jun 2015

https://www.youtube.com/watch?v=TRcReyRYIMg

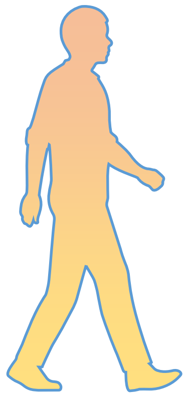
Scrum

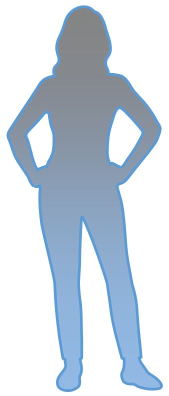
- Scrum is a lightweight, iterative method for managing software development in a complex and changing environment

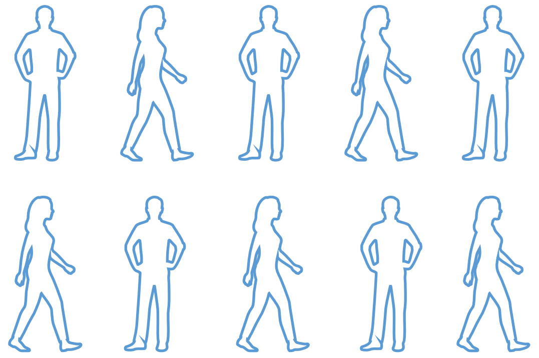
- It has a cyclical nature – the "sprint", and processes within it, repeat over and over



Scrum – Accountabilities







Developers, QA, UI designers, …

Product Owner

- knows what needs to be built, and in what sequence
- accountable for the product backlog and maximising business value
- represents the business needs of the product

Scrum Master

- accountable for removing impediments the team face
- responsible for enacting scrum values and practices
- **NOT** a traditional project manager

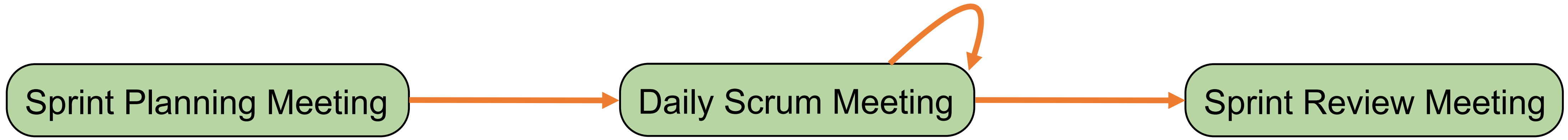
Team

- cross-functional
- self-organising
- members are full-time
- membership can only change between sprints

Scrum – Sprints

- Sprints are fixed-length (e.g. 2-4 weeks) iterations that aim to achieve some agreed-upon outcome
  - e.g. some software functionality – tested, integrated, documented

- Requirements are frozen for the length of a sprint



- agree sprint goal based on priorities set by Product Owner
- select product backlog items that contribute to it
- <8hrs for a 4 week sprint

- short (~15 mins); facilitated by scrum master
- identify impediments to progress
- **NOT** about finding out who is behind schedule

- held once at the end
- new functionality demonstrated to Product Owner
- discuss impact of incomplete work

Scrum Board (or Task Board)

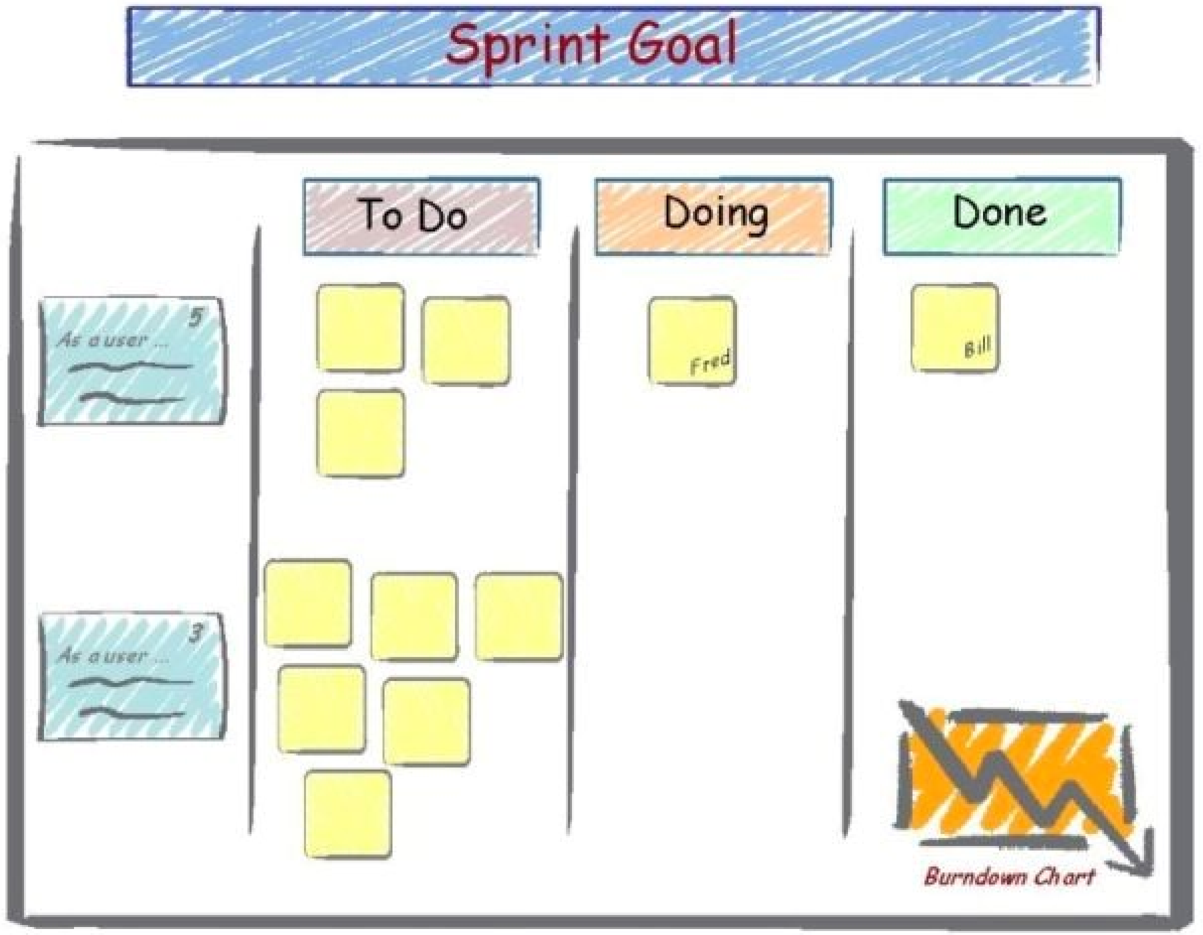
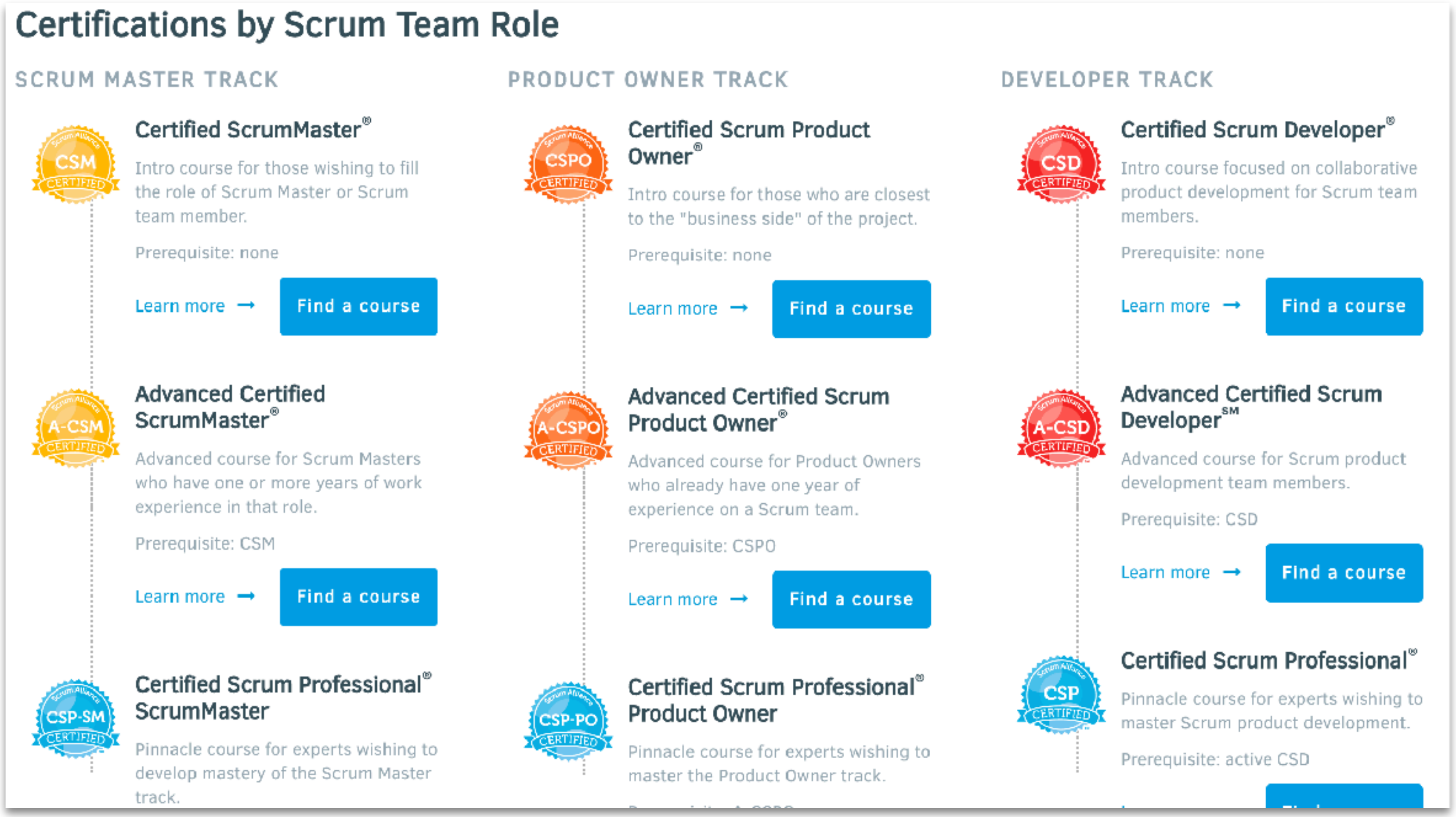


image courtesy of yorkesoftware.com/2015/01/30/explaining-the-taskboard/

Scrum – Certification

- The Scrum Alliance provides a number of certification routes for those keen to develop their agile credentials



Certifications by Scrum Team Role

SCRUM MASTER TRACK

Certified ScrumMaster®
Intro course for those wishing to fill the role of Scrum Master or Scrum team member.
Prerequisite: none
Learn more → Find a course

Advanced Certified ScrumMaster®
Advanced course for Scrum Masters who have one or more years of work experience in that role.
Prerequisite: CSM
Learn more → Find a course

Certified Scrum Professional® ScrumMaster
Pinnacle course for experts wishing to develop mastery of the Scrum Master track.

PRODUCT OWNER TRACK

Certified Scrum Product Owner®
Intro course for those who are closest to the "business side" of the project.
Prerequisite: none
Learn more → Find a course

Advanced Certified Scrum Product Owner®
Advanced course for Product Owners who already have one year of experience on a Scrum team.
Prerequisite: CSPO
Learn more → Find a course

Certified Scrum Professional® Product Owner
Pinnacle course for experts wishing to master the Product Owner track.

DEVELOPER TRACK

Certified Scrum Developer®
Intro course focused on collaborative product development for Scrum team members.
Prerequisite: none
Learn more → Find a course

Advanced Certified Scrum Developer℠
Advanced course for Scrum product development team members.
Prerequisite: CSD
Learn more → Find a course

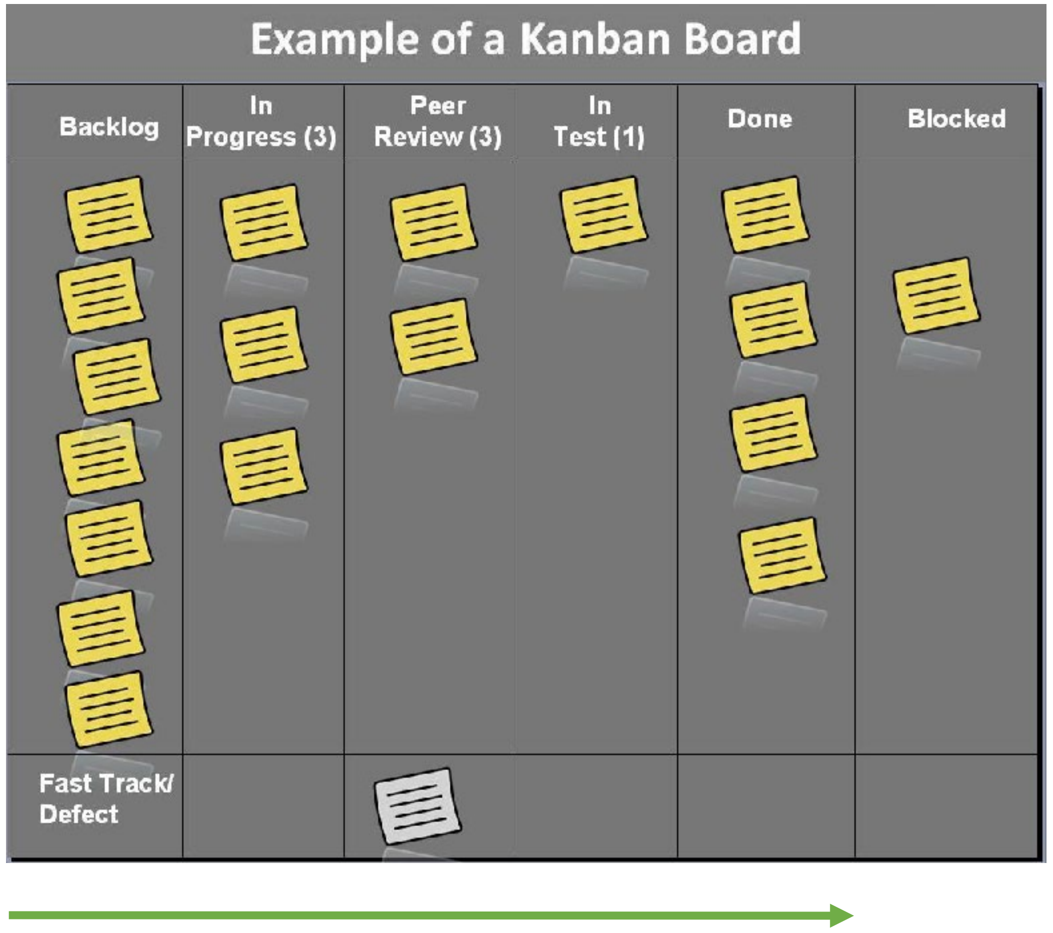
Certified Scrum Professional®
Pinnacle course for experts wishing to master Scrum product development.
Prerequisite: active CSD

https://www.scrumalliance.org/

Scrum – Impact

- Scrum has turned the general idea of iterative development into a precise discipline
  - Codified goals, duration, and management of iterations (sprints)
- Savvy marketing operation – certifications turn scrum learners into scrum supporters
- Scrum/sprints have quickly become the industry standard iteration model – even beyond software development

Kanban



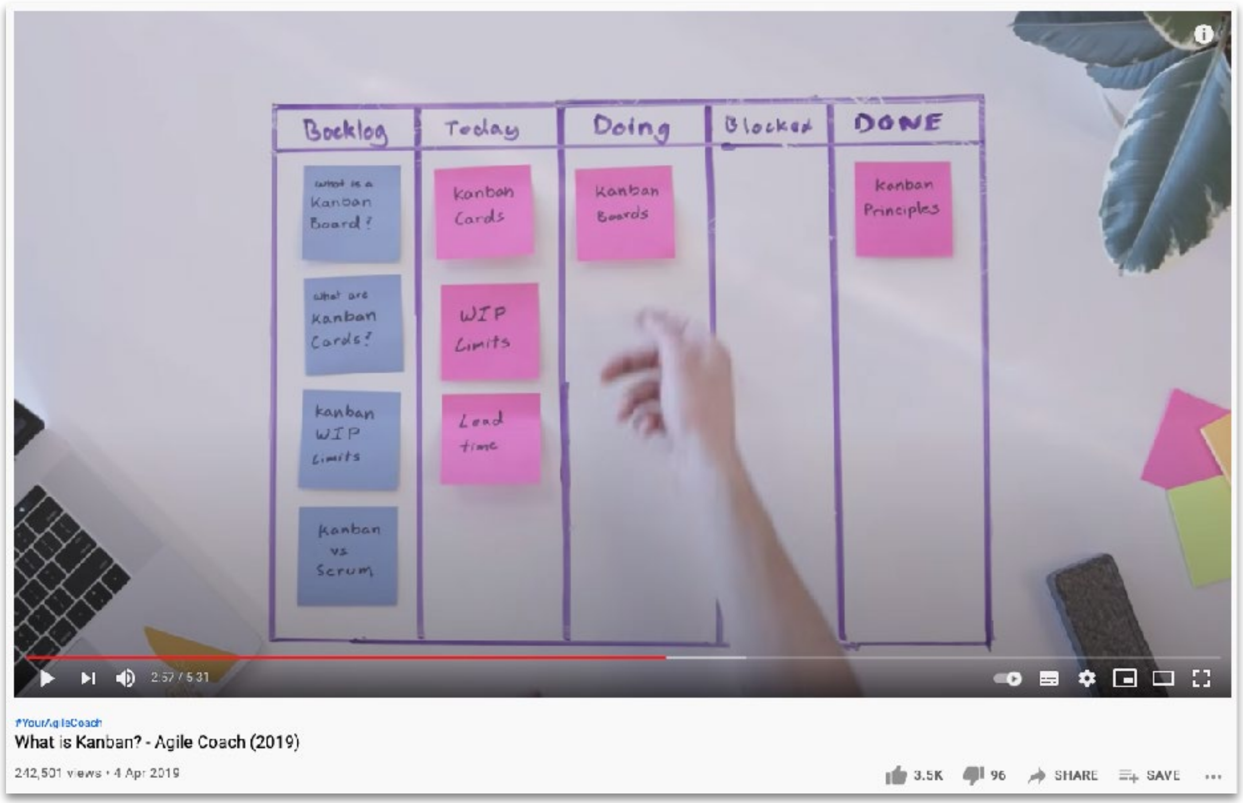
"kanban"

- *Japanese word meaning (roughly) "large visual board"*

- *idea: visualising a team's workflow helps identify potential waste / constraints*

- *NB: 'kanban board' is used throughout the whole project; a 'scrum board' is cleared after every sprint*

Kanban

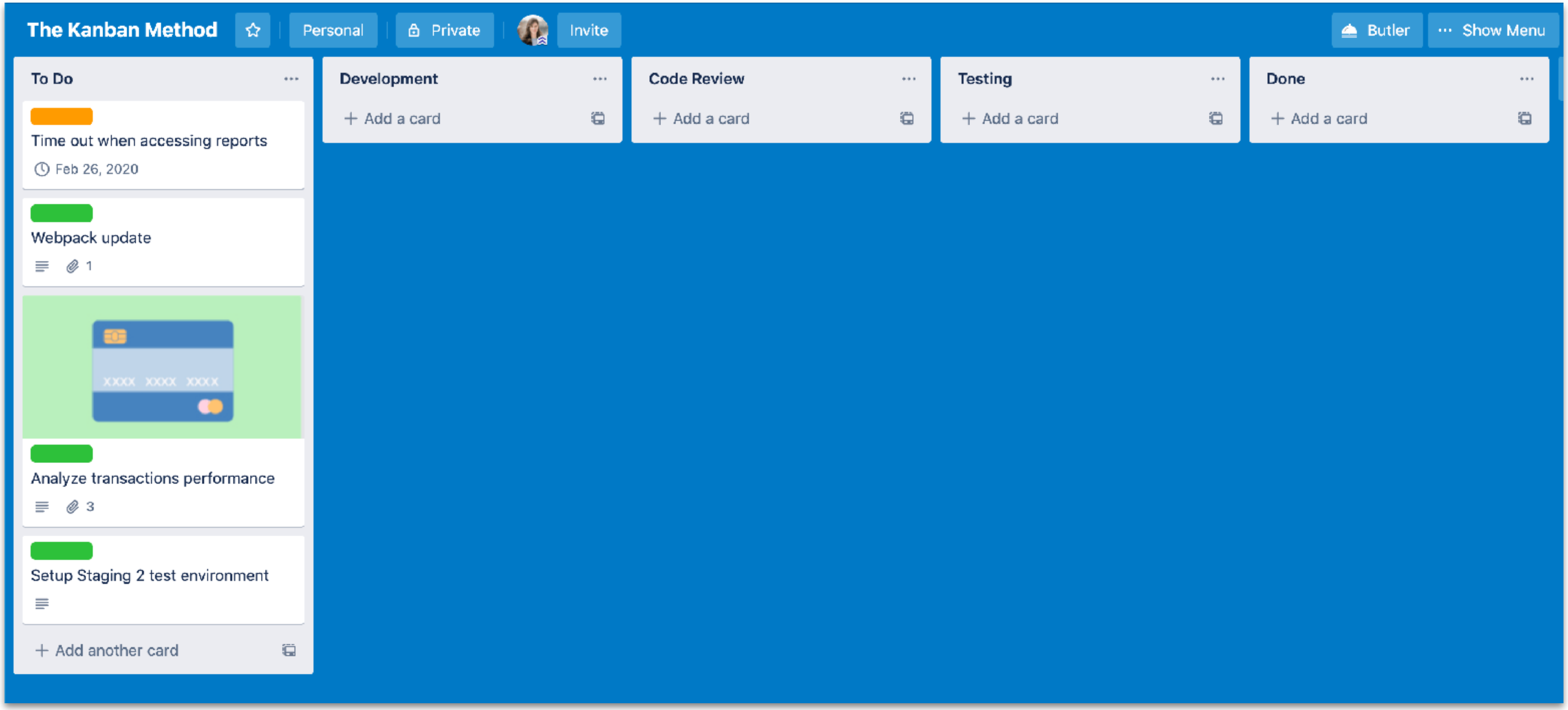


https://www.youtube.com/watch?v=iVaFVa7HYj4&t=116s

Kanban

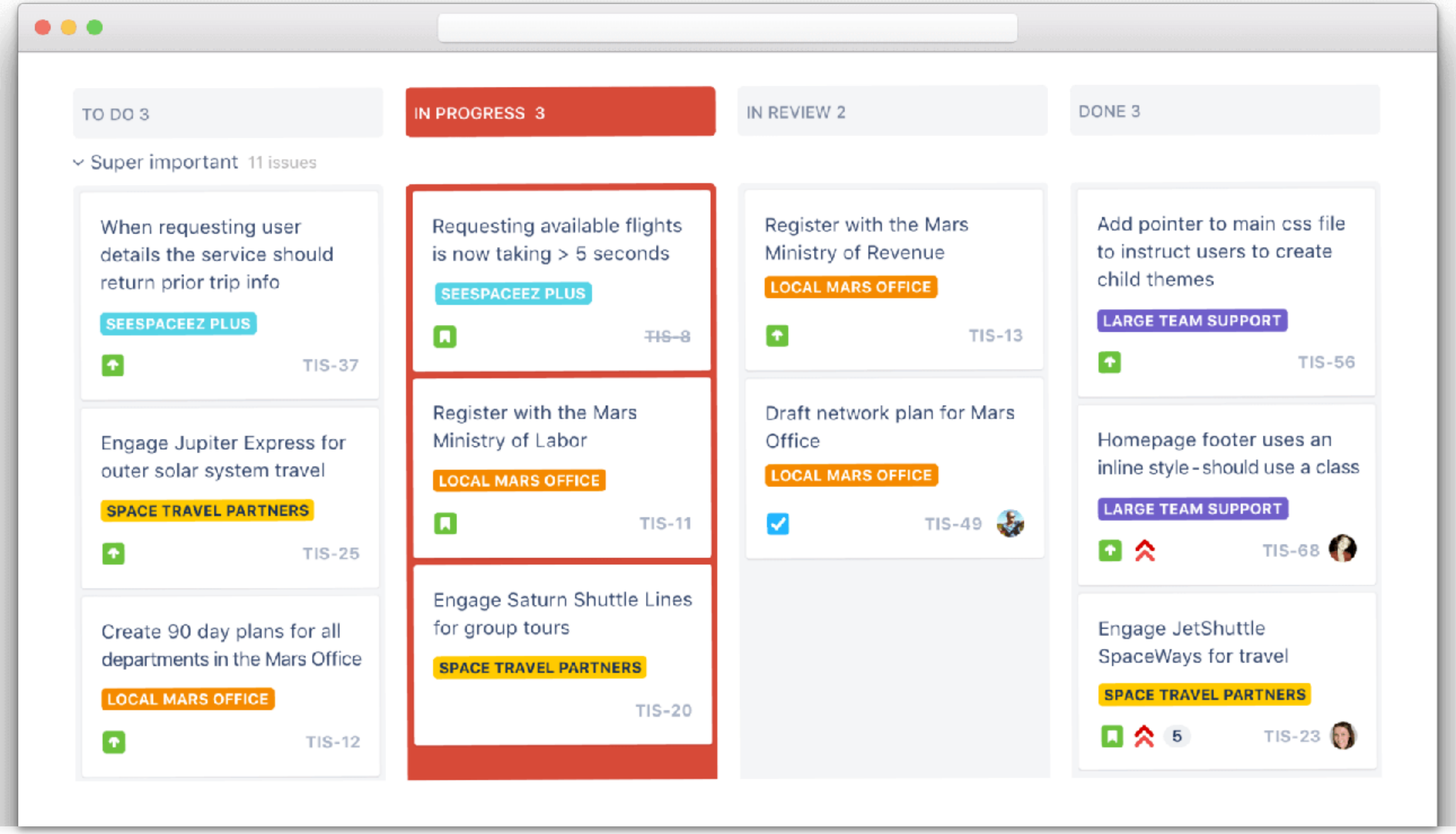
- Work-in-progress (WIP) doesn't deliver value until deployed
- Limiting / minimising WIP makes it easier to identify inefficiencies and constraints in a team's workflow
- Kanban: clearly visualise WIP and match to team's capacity
- Simple, but effective approach to help teams coordinate

Kanban – Virtual Boards



https://trello.com/en

Kanban – Virtual Boards



TO DO 3

Super important 11 issues

When requesting user details the service should return prior trip info
SEESPACEEZ PLUS
TIS-37

Engage Jupiter Express for outer solar system travel
SPACE TRAVEL PARTNERS
TIS-25

Create 90 day plans for all departments in the Mars Office
LOCAL MARS OFFICE
TIS-12

IN PROGRESS 3

Requesting available flights is now taking > 5 seconds
SEESPACEEZ PLUS
~~TIS-8~~

Register with the Mars Ministry of Labor
LOCAL MARS OFFICE
TIS-11

Engage Saturn Shuttle Lines for group tours
SPACE TRAVEL PARTNERS
TIS-20

IN REVIEW 2

Register with the Mars Ministry of Revenue
LOCAL MARS OFFICE
TIS-13

Draft network plan for Mars Office
LOCAL MARS OFFICE
TIS-49

DONE 3

Add pointer to main css file to instruct users to create child themes
LARGE TEAM SUPPORT
TIS-56

Homepage footer uses an inline style - should use a class
LARGE TEAM SUPPORT
TIS-68

Engage JetShuttle SpaceWays for travel
SPACE TRAVEL PARTNERS
5
TIS-23

https://www.atlassian.com/software/jira

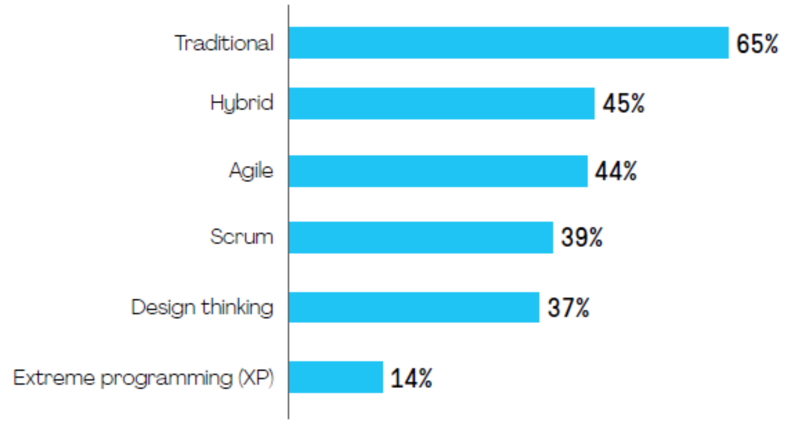
Scrum vs. Kanban

- Kanban is a different framework to scrum, but it shares some similar foundational principles (e.g. minimising WIP)

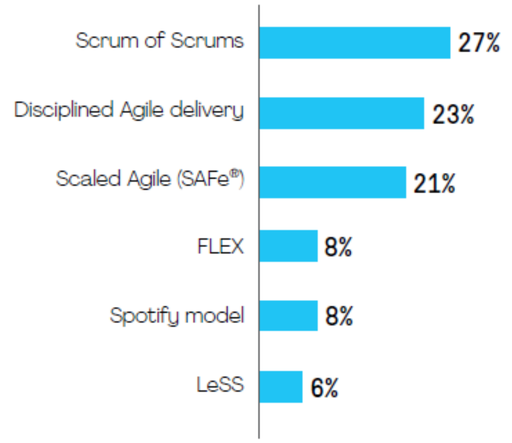
	Scrum	Kanban
Cadence	Fixed-length sprints	Continuous flow
Release methodology	End of each sprint	Continuous delivery
Accountabilities / Roles	Product owner, scrum master, scrum team	No required roles
Key metrics	Velocity	Lead time, cycle time, WIP
Change philosophy	Requirements frozen during sprints	Requirements can change at any point
Visualisation	Scrum board cleared after every sprint	Kanban board used throughout project

PMI – Pulse of the Profession 2021

Q: How often does your organization use these approaches? (% always/often)

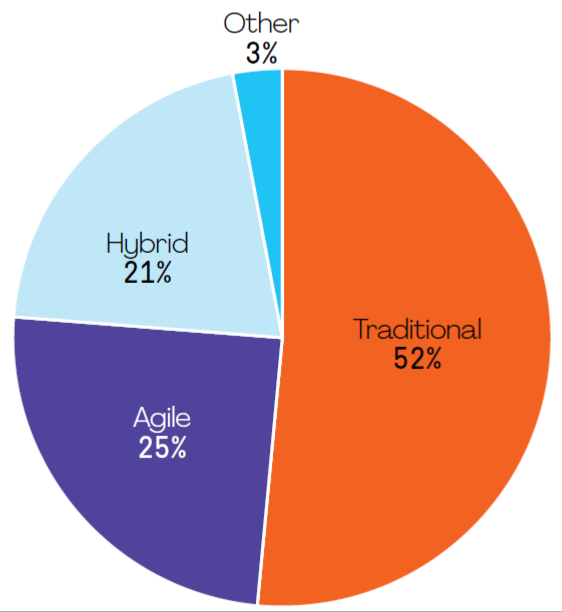


Q: How often are the following agile and/or hybrid approaches used? (% always/often among agile/hybrid users)



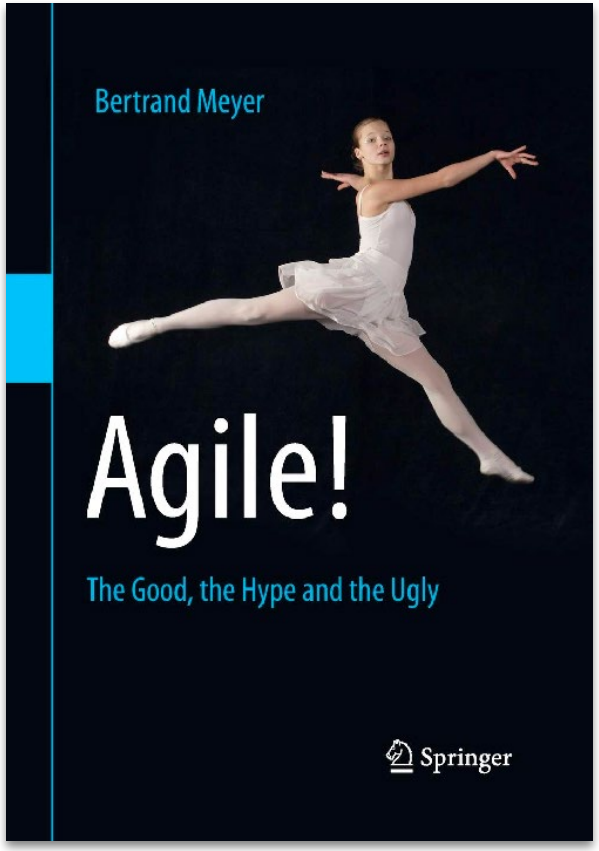
"Organizations shouldn't mandate any one framework or think that one size fits all," says Sahar Kanani, PMP, director of program management at health technology company MacroHealth, Vancouver, Canada. "As much as I'm an advocate for agile, I also believe you couldn't land on the moon or even build a bridge without waterfall."

Q: In your estimation, what percentage of the projects completed within your organization in the past 12 months used the following types of approaches?

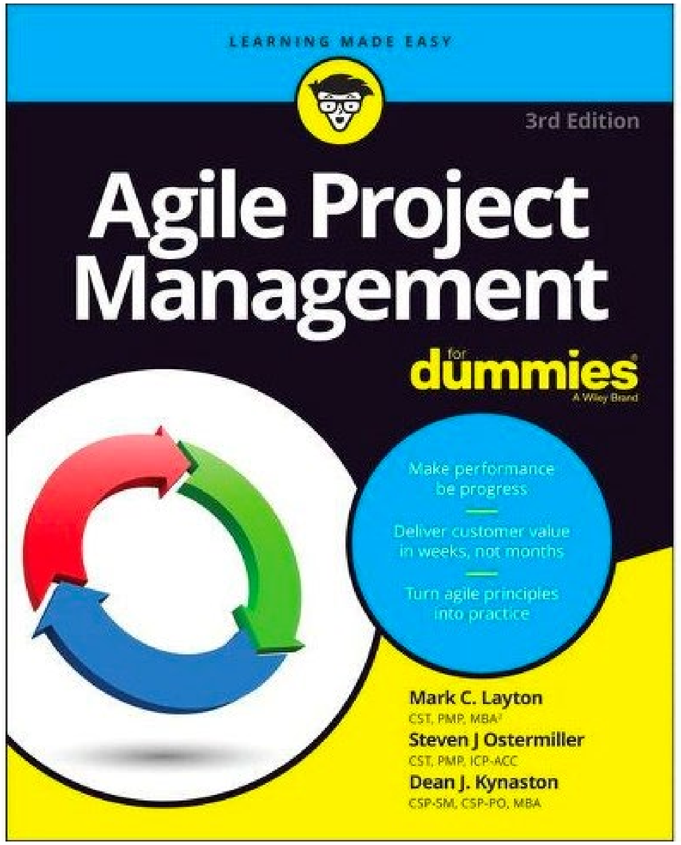


https://www.pmi.org/learning/thought-leadership/pulse/pulse-of-the-profession-2021

References / Additional Reading



(at least chapter 11)



(ignore the title!)