Software Project Management – IS212

Software Development Process

Software Development Processes

Objectives

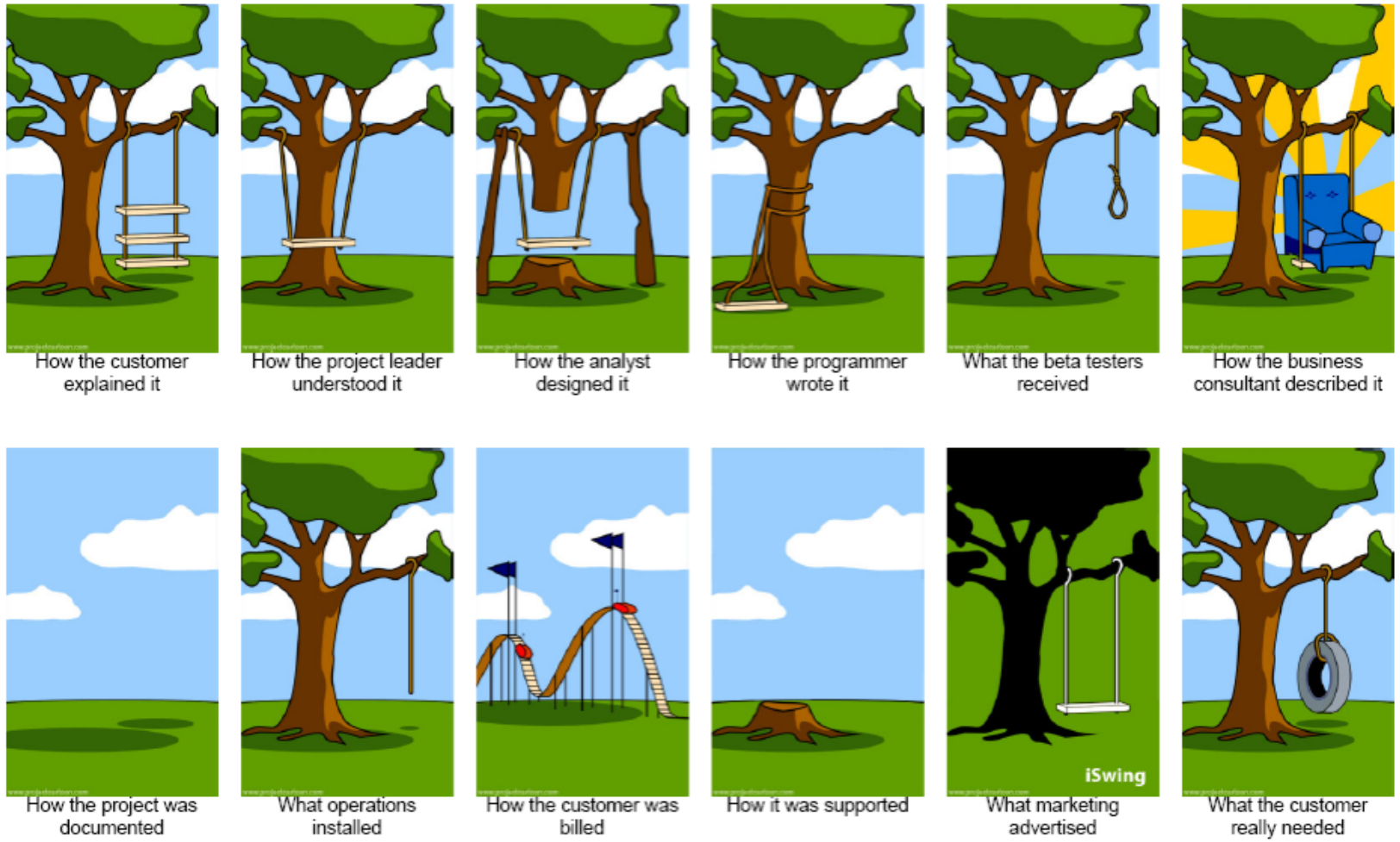
On completing this module, you will be able to:

- 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

A classic comic…



Software Development Methods

Traditional

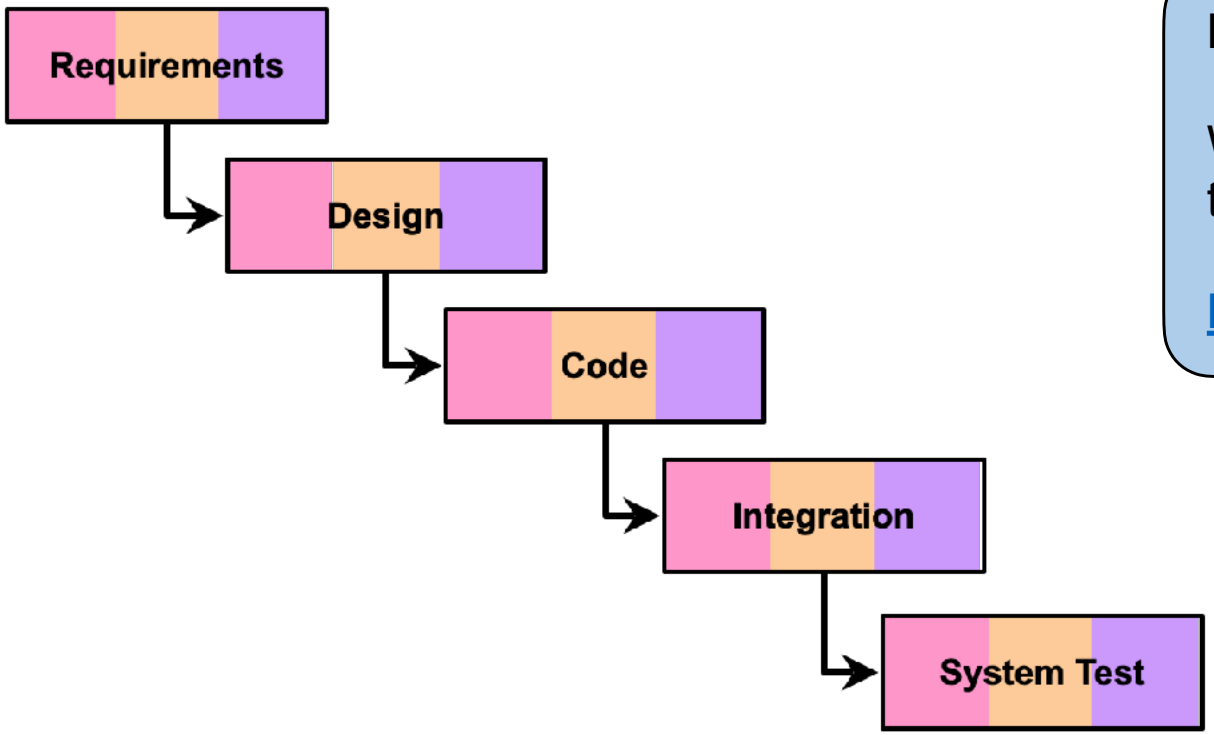
- Waterfall
- RUP
- Spiral
- Prototyping

Agile

- Extreme Programming
- Scrum
- Kanban

Waterfall

Idea: break software development into linear, sequential phases



Padlet Exercise

What are some **pros** and **cons** of this method?

https://padlet.com/XXX/YYY

Waterfall – Discussion

Real-world software projects are iterative

Software Development Methods

Traditional

- Waterfall
- RUP
- Spiral
- Prototyping

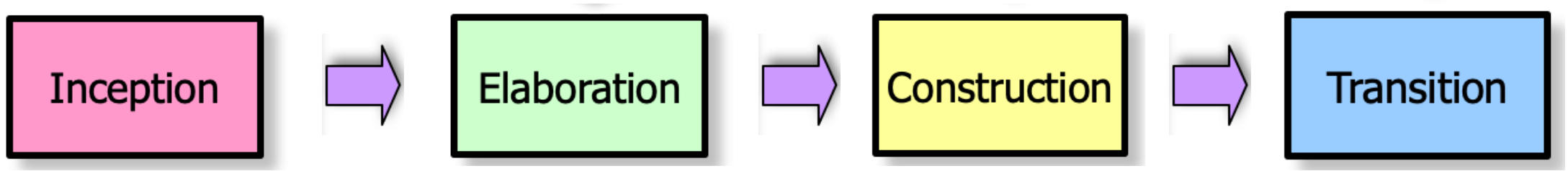
iterative traditional methods —>

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

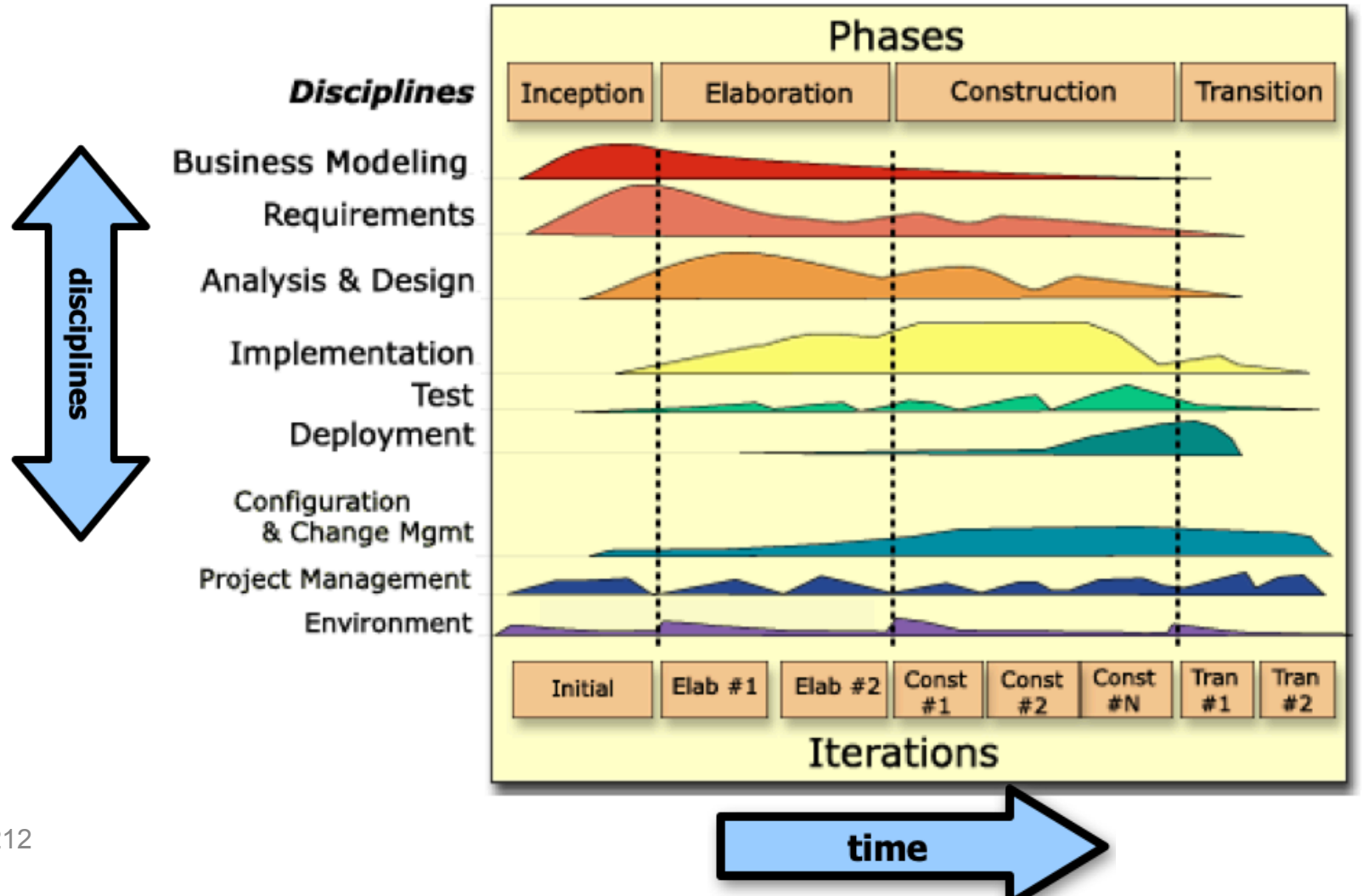
actually ***build*** the product

validate and ***deploy*** the product

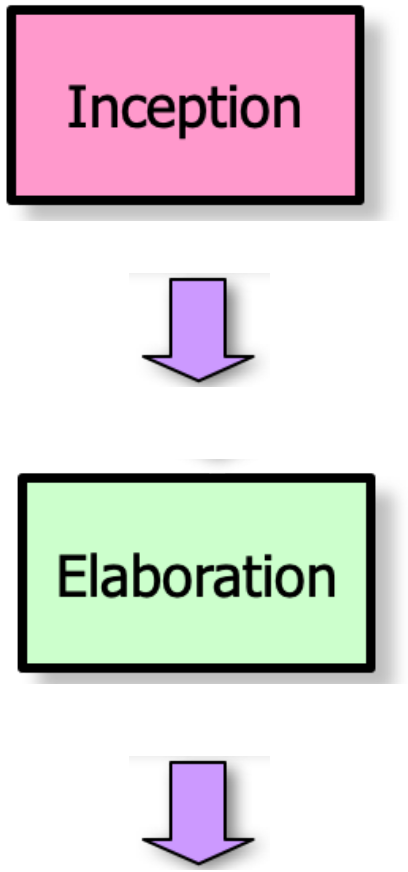
- Various 'engineering disciplines' then cut across all four phases
  - *e.g. business modelling, requirements, implementation, testing*

Rational Unified Process (RUP)

- Unlike waterfall – it explicitly encourages iterations within phases



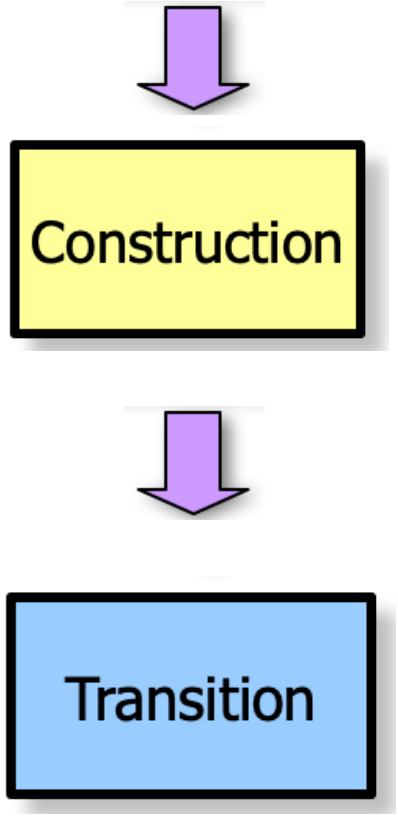
RUP phases – possible outcomes



- 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

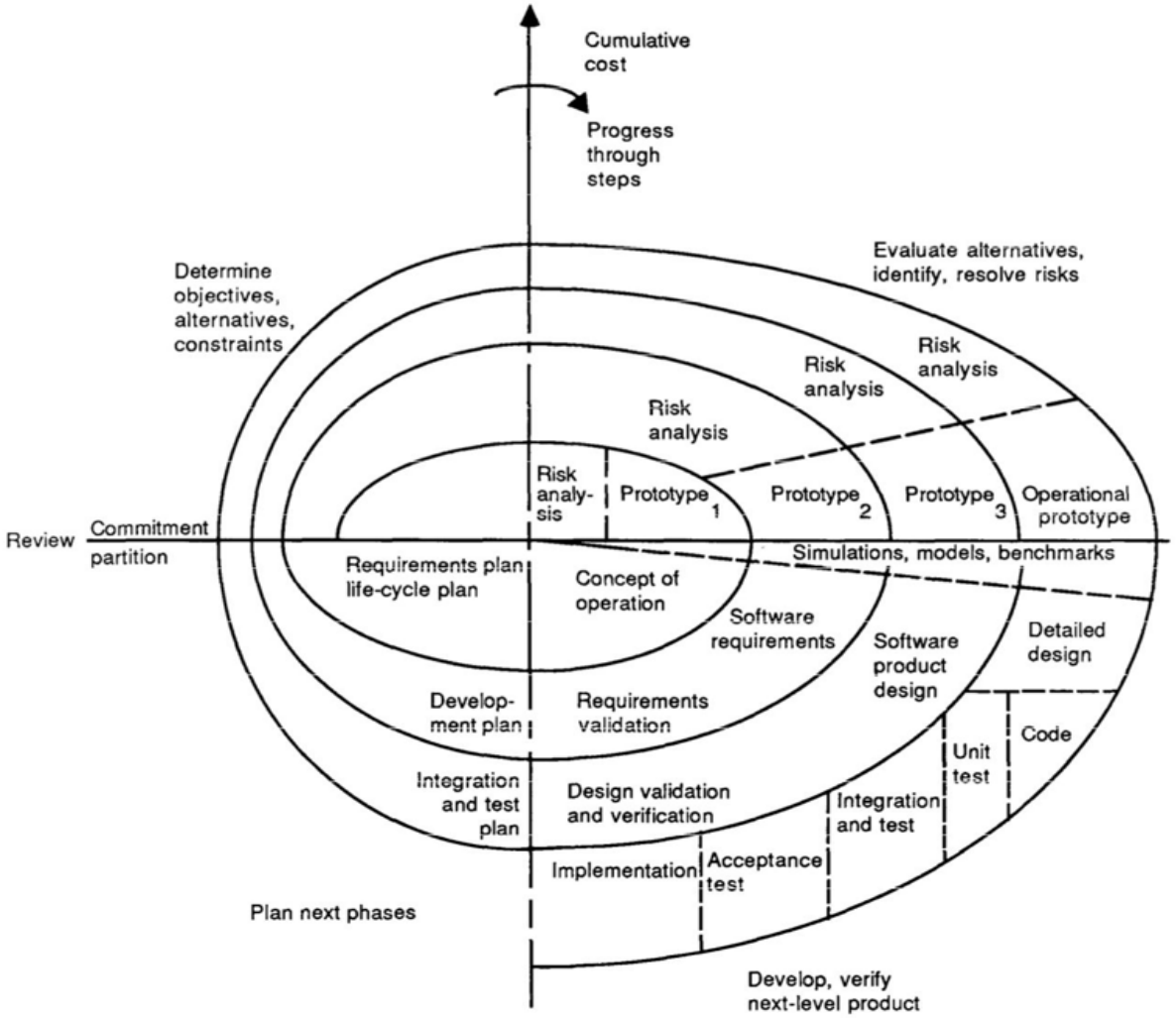
- Training end users / maintainers
- Evaluate against ‘Inception’ requirements
- ‘Post-mortem’ project analysis

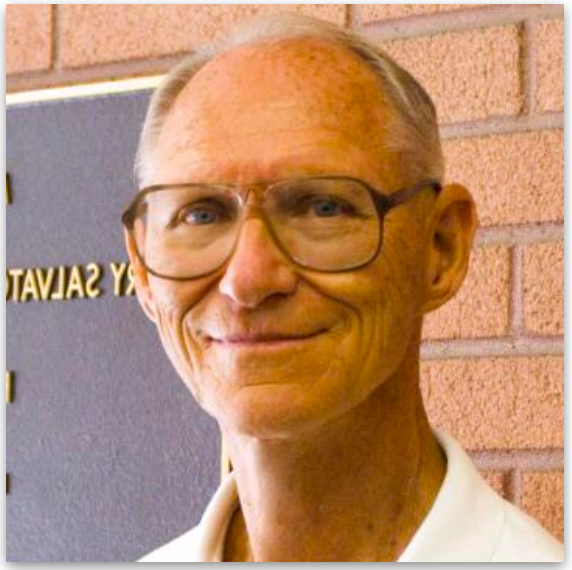
RUP – Discussion

- 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!)*

Spiral

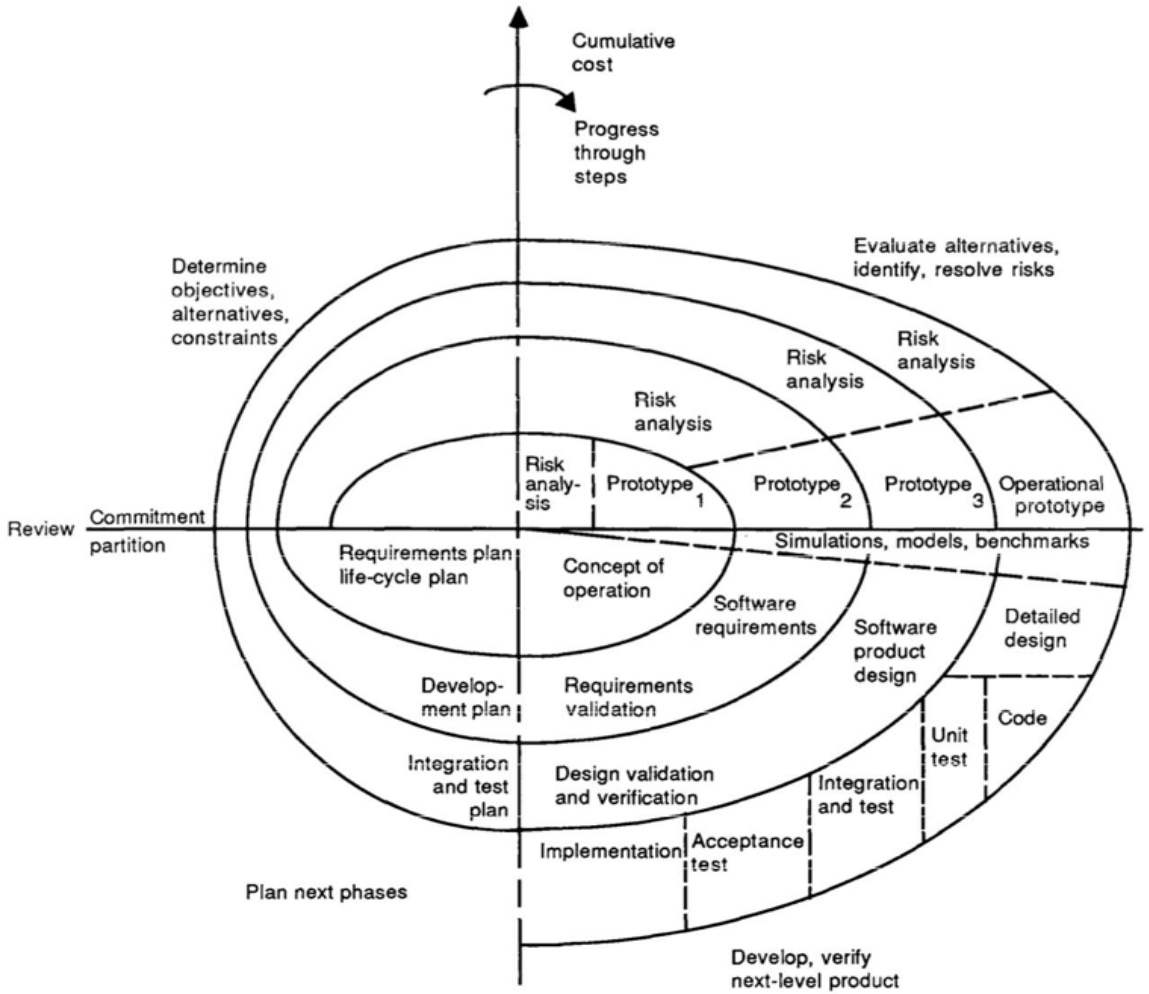
- Spiral – a classic risk-driven method proposed by Barry Boehm





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

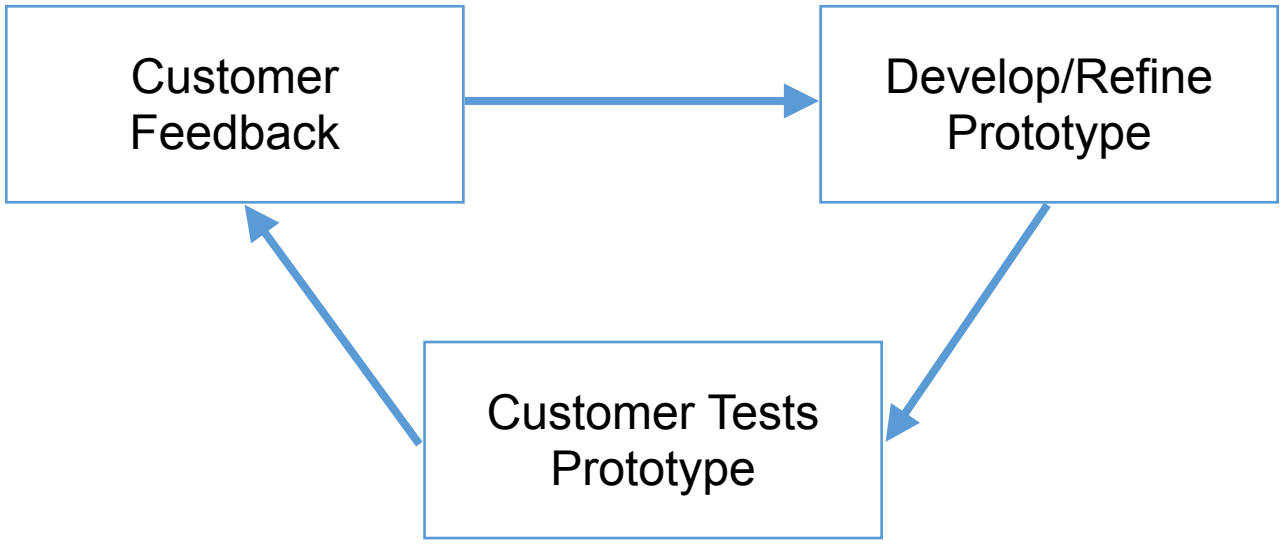


Discussion: what are the pros and cons of the spiral method?

Prototyping Method

- This final method focuses on the activity of prototyping itself

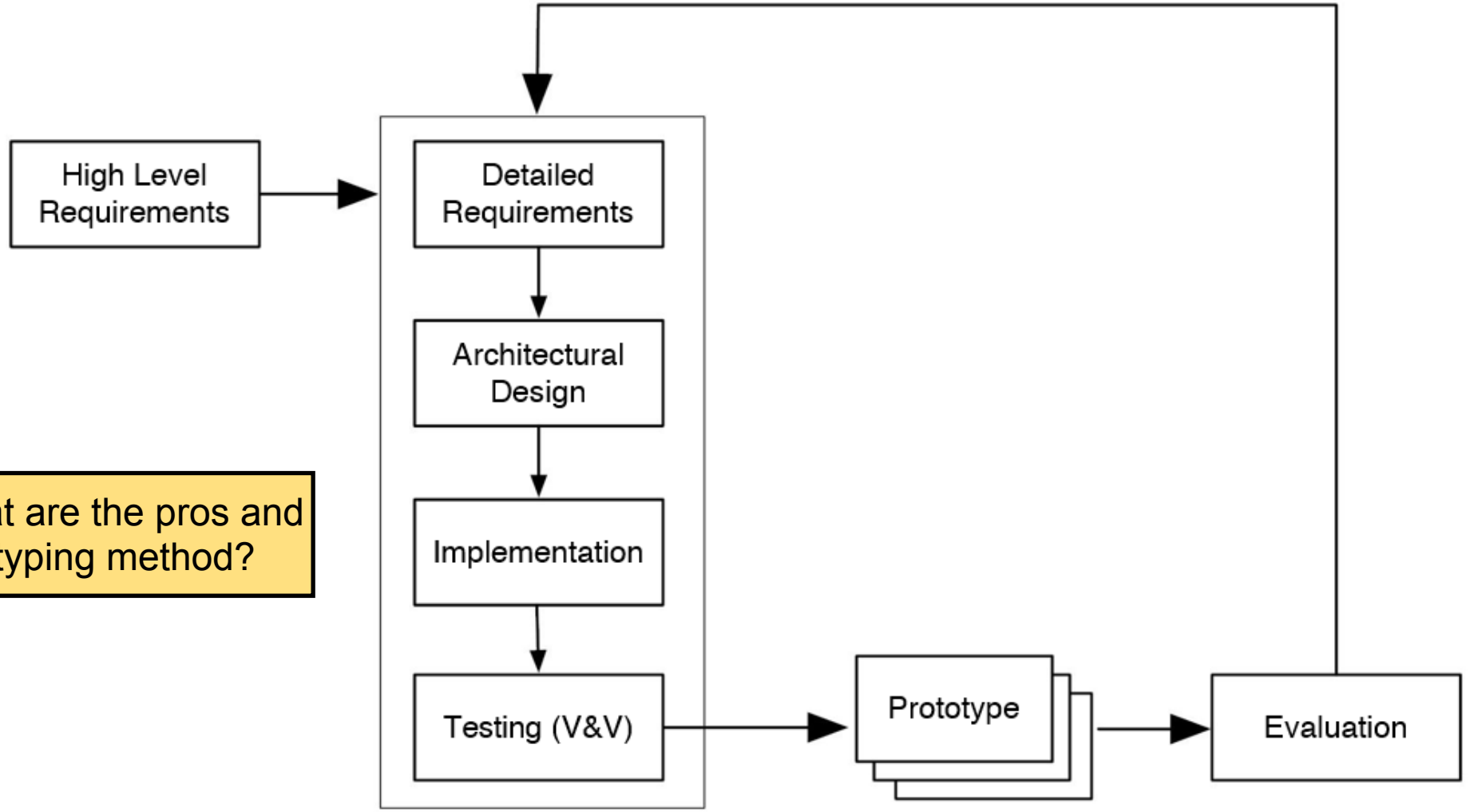
- Useful for when customers do not know their exact requirements
  - Roughly:



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

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

Prototyping Method *(more detail)*



Discussion: what are the pros and cons of the prototyping method?

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		comprehensive documentation
Customer collaboration		contract negotiation
Responding to change		following a plan

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

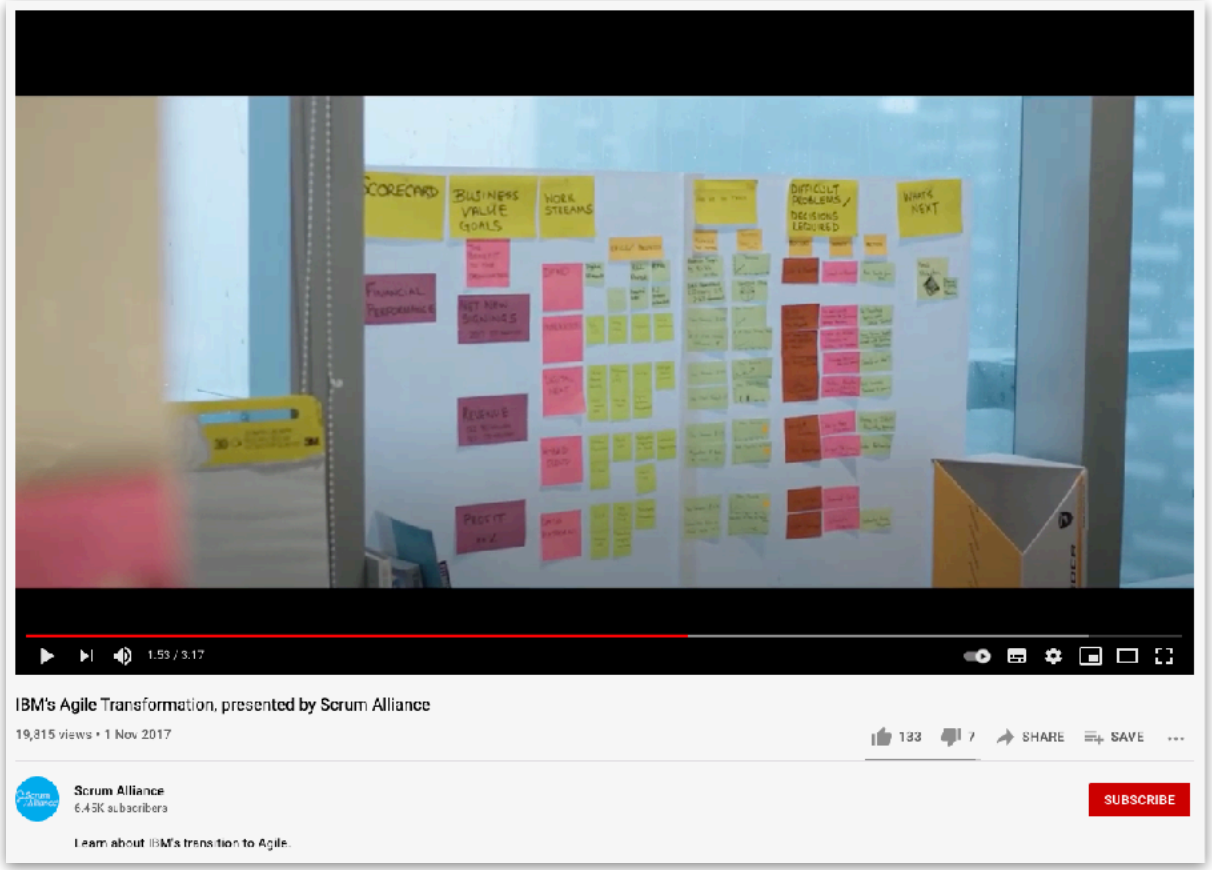
Agile – IBM Video

Padlet Exercise

Why has IBM adopted agile methods?

What 'tools' do you see the teams using?

https://padlet.com/XXX/YYY



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

Highly iterative and flexible

Working software delivered frequently

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

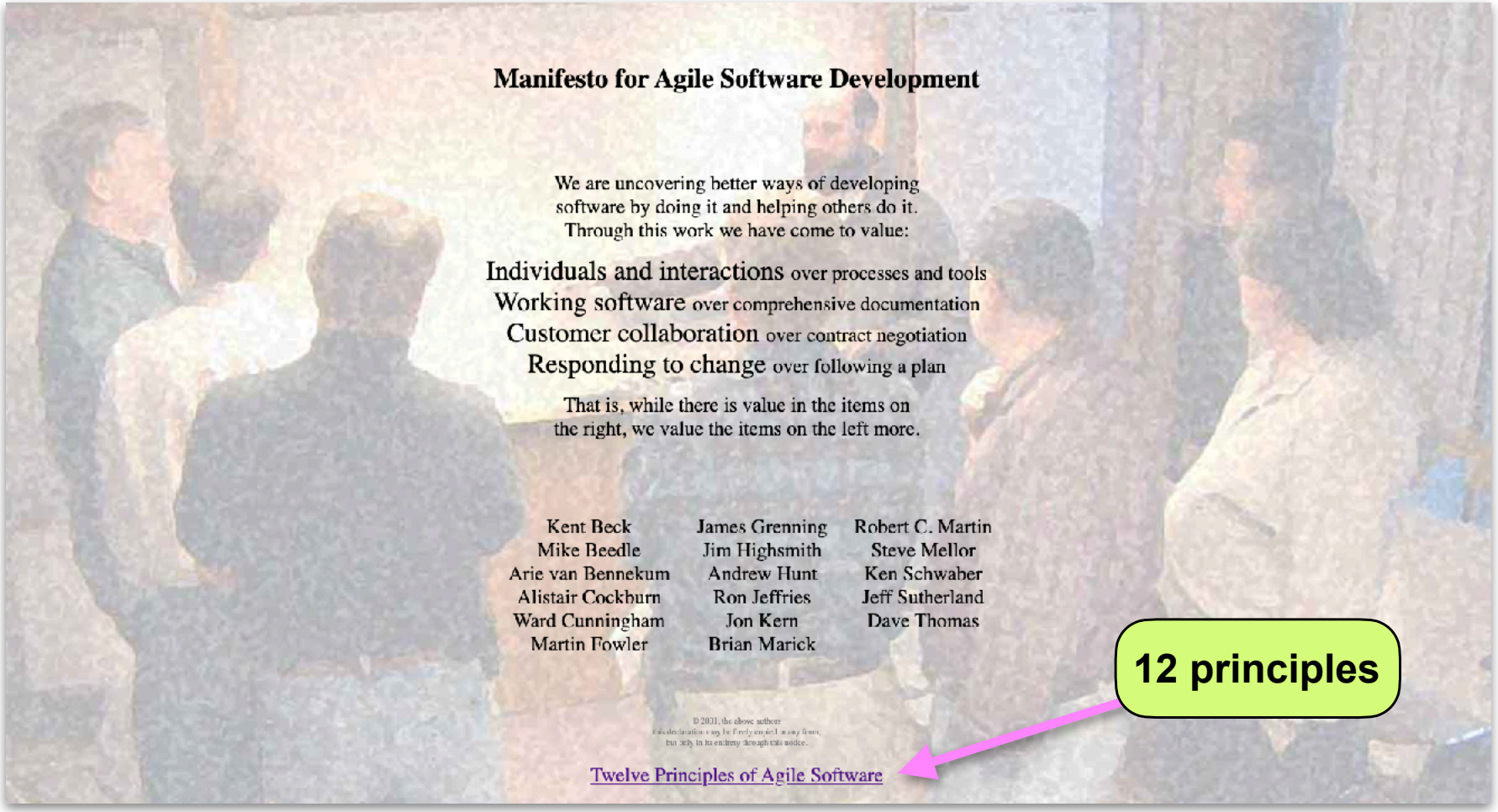
Agile – Key Characteristics

Late changes in requirements – no problem

Close, daily co-operation

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

Agile Manifesto (2001)



https://agilemanifesto.org/

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

Agile principles

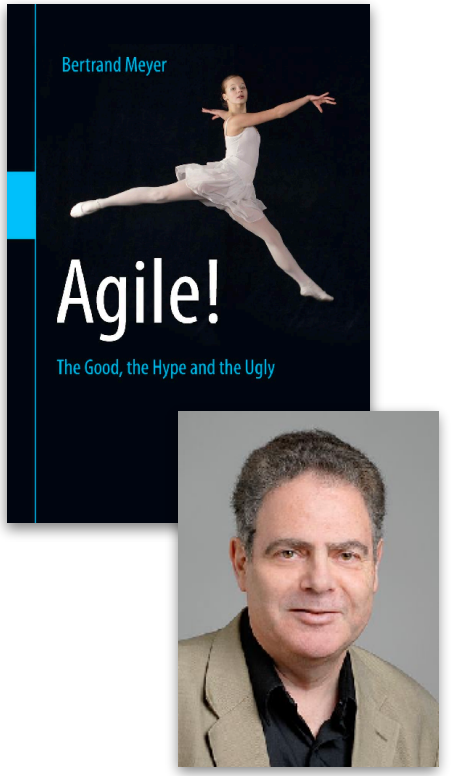
Organizational

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

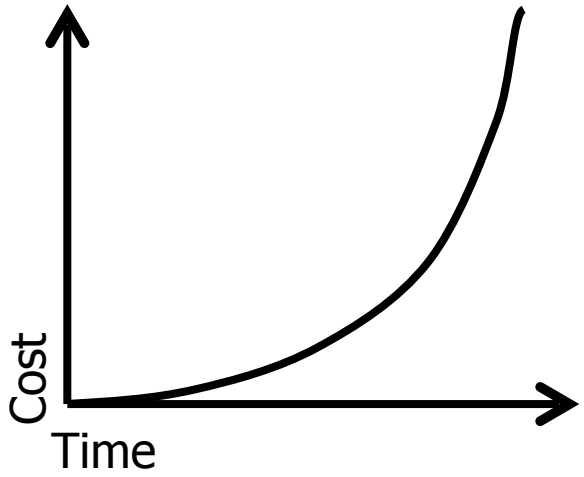
Agile

- Extreme Programming
- Scrum
- Kanban

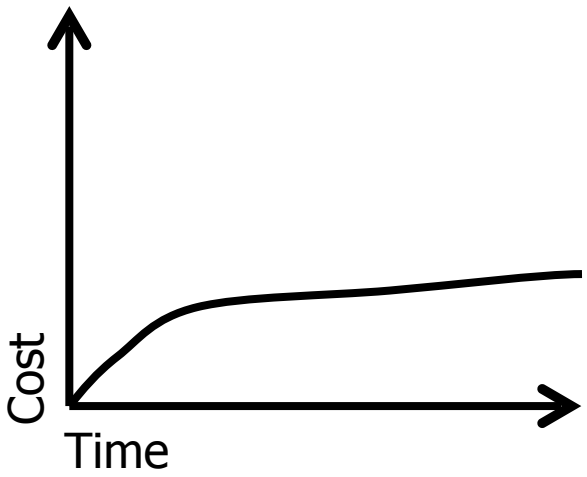
Extreme Programming (XP)

- Premise of extreme programming (XP)

Traditional
- upfront design
- code late
- release when ‘done’



XP
- code early
- release early
- continuous design



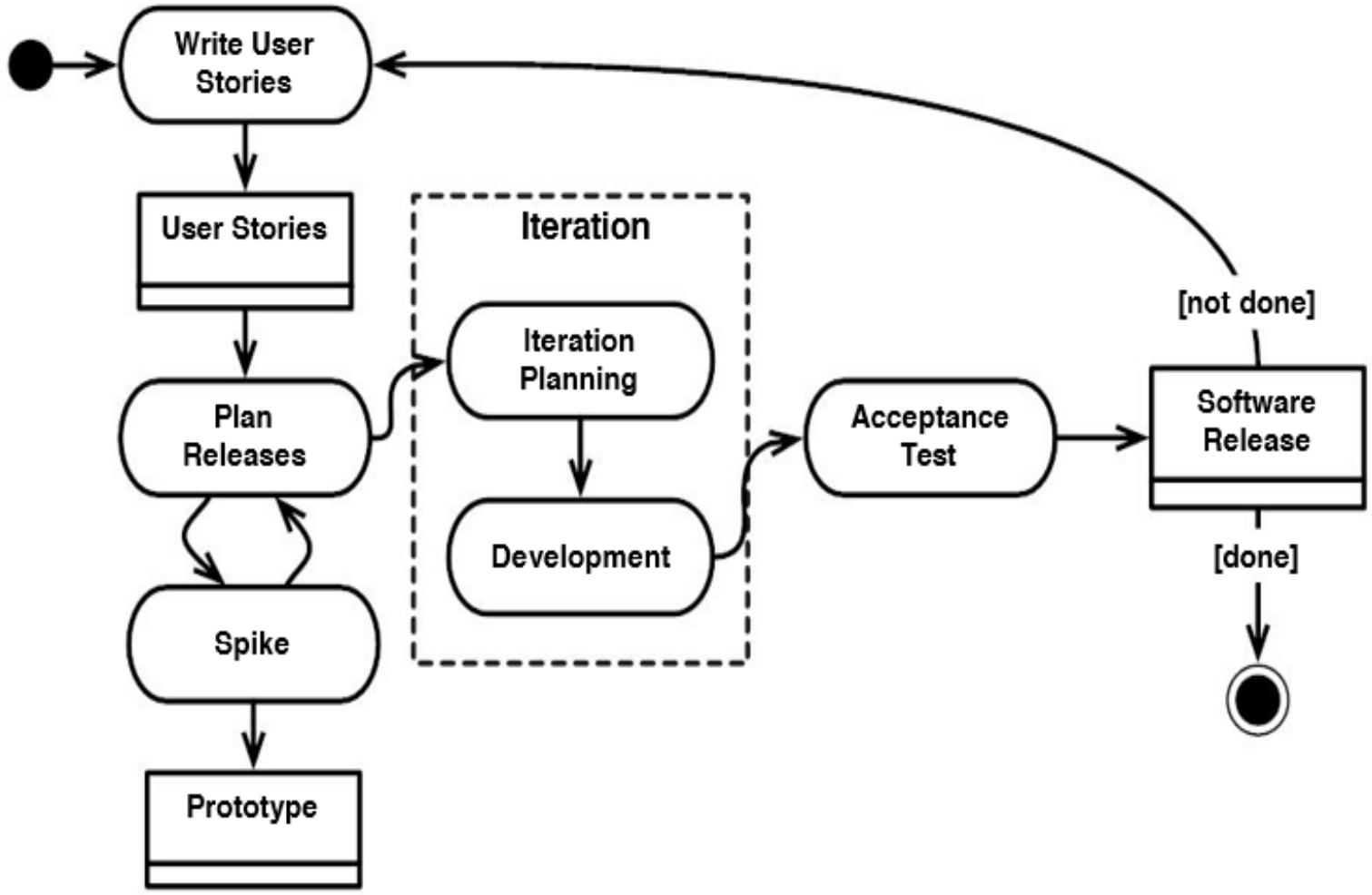
Extreme Programming (XP)

- 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
Plan Releases
Spike
Prototype
Iteration
Iteration Planning
Development
Acceptance Test
[not done]
Software Release
[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

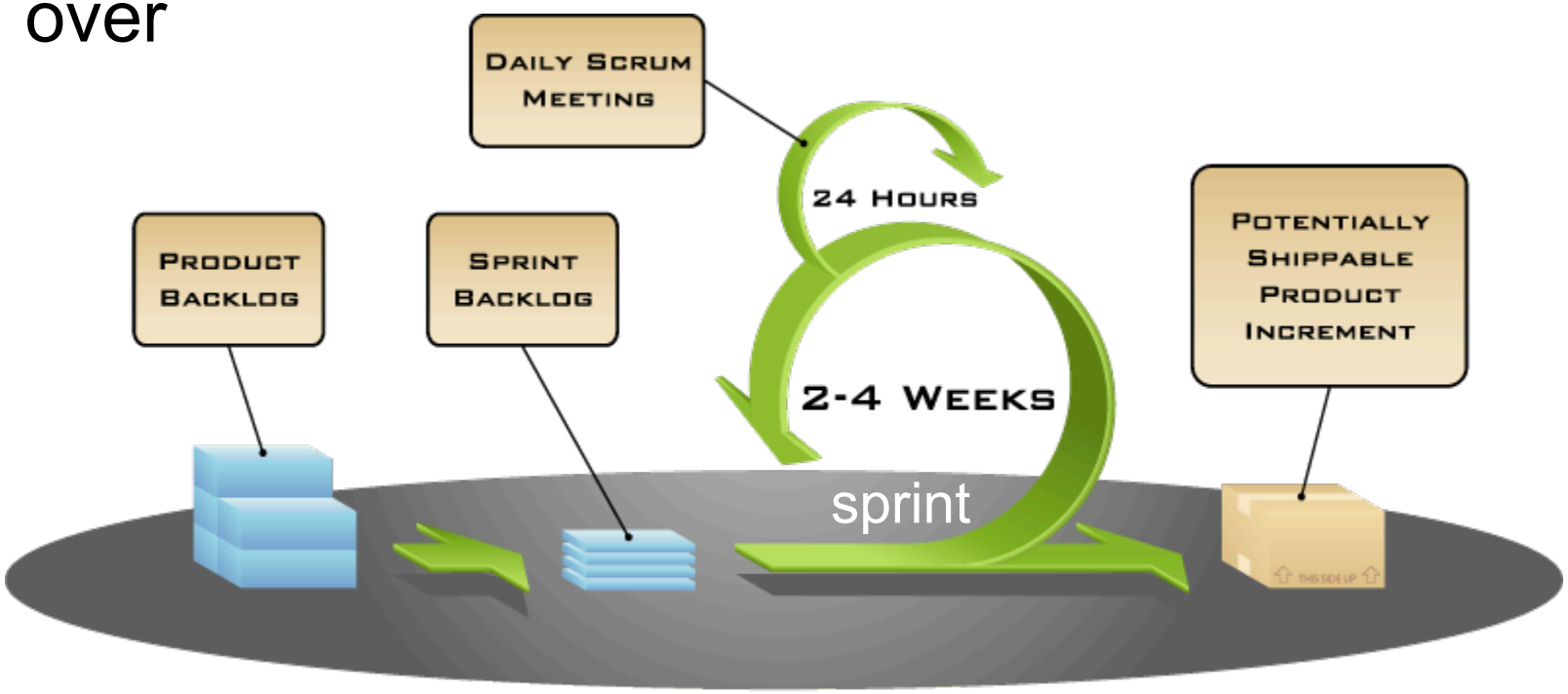
Scrum

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

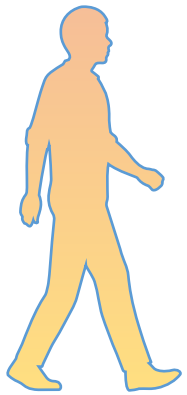
Scrum

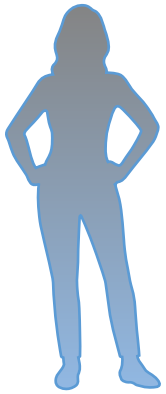
- Scrum is a lightweight, iterative method for managing software development in a complex and changing environment

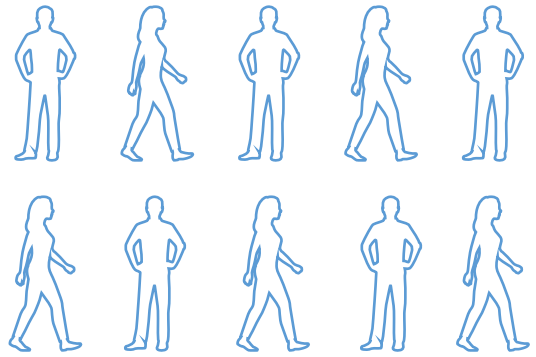
- It has a cyclical nature – the "sprint", and processes within it, repeat over and over



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

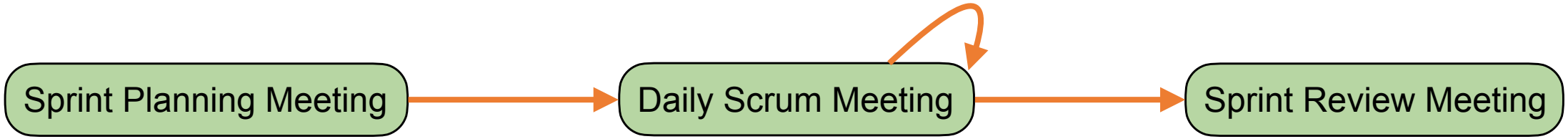
Scrum Team

- 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

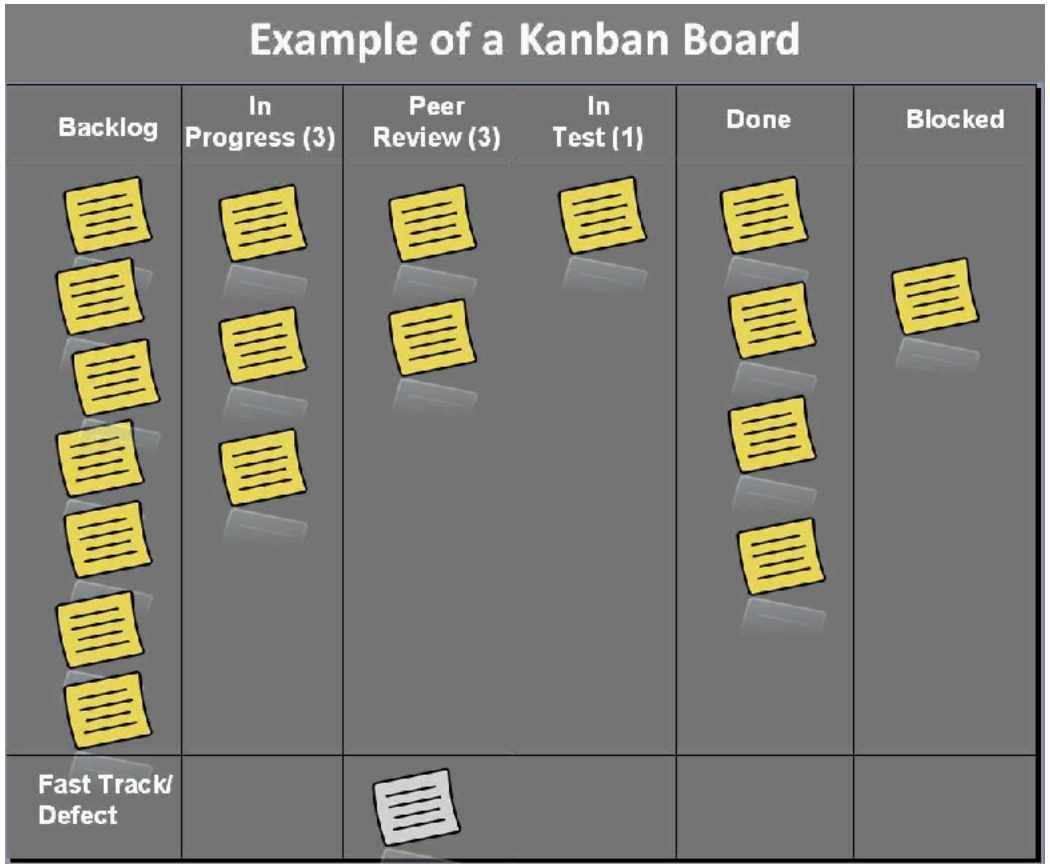
- The Scrum Alliance provides a number of certification routes for those keen to develop their agile credentials

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



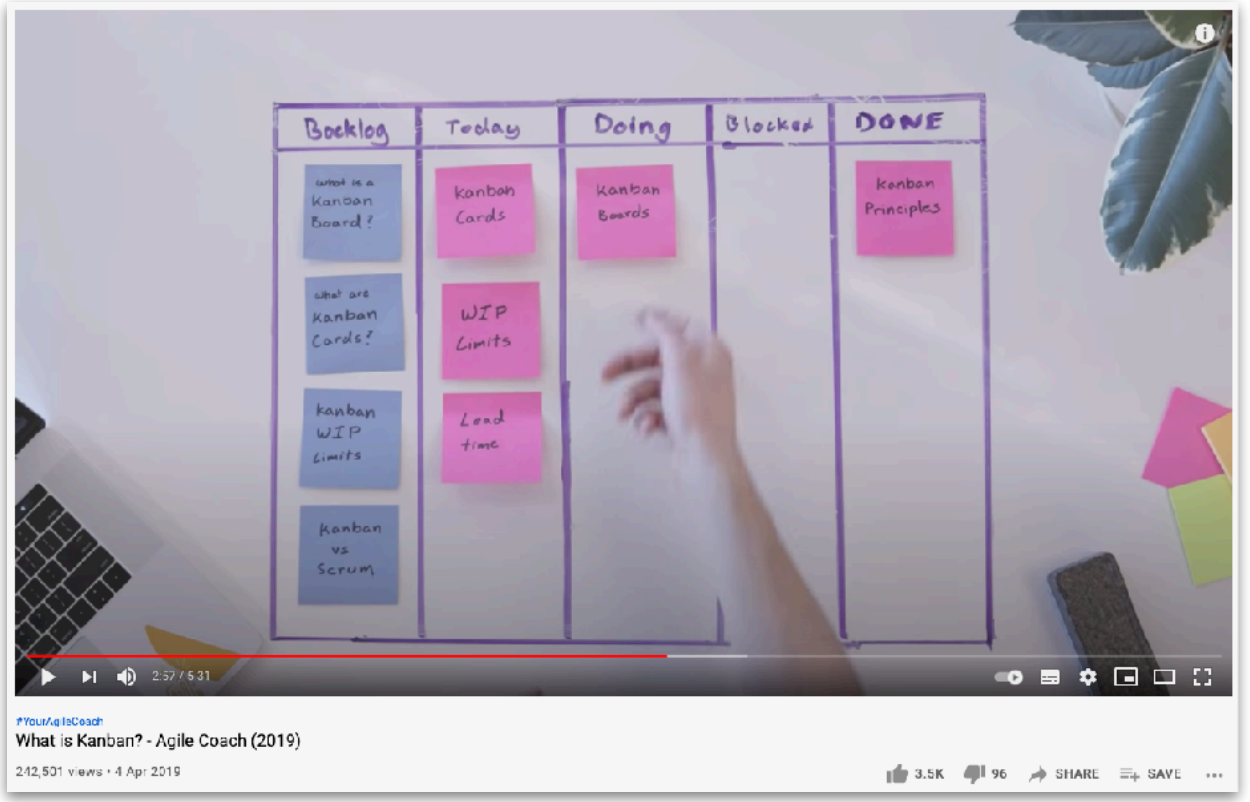
*"**kanban**"*

- *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

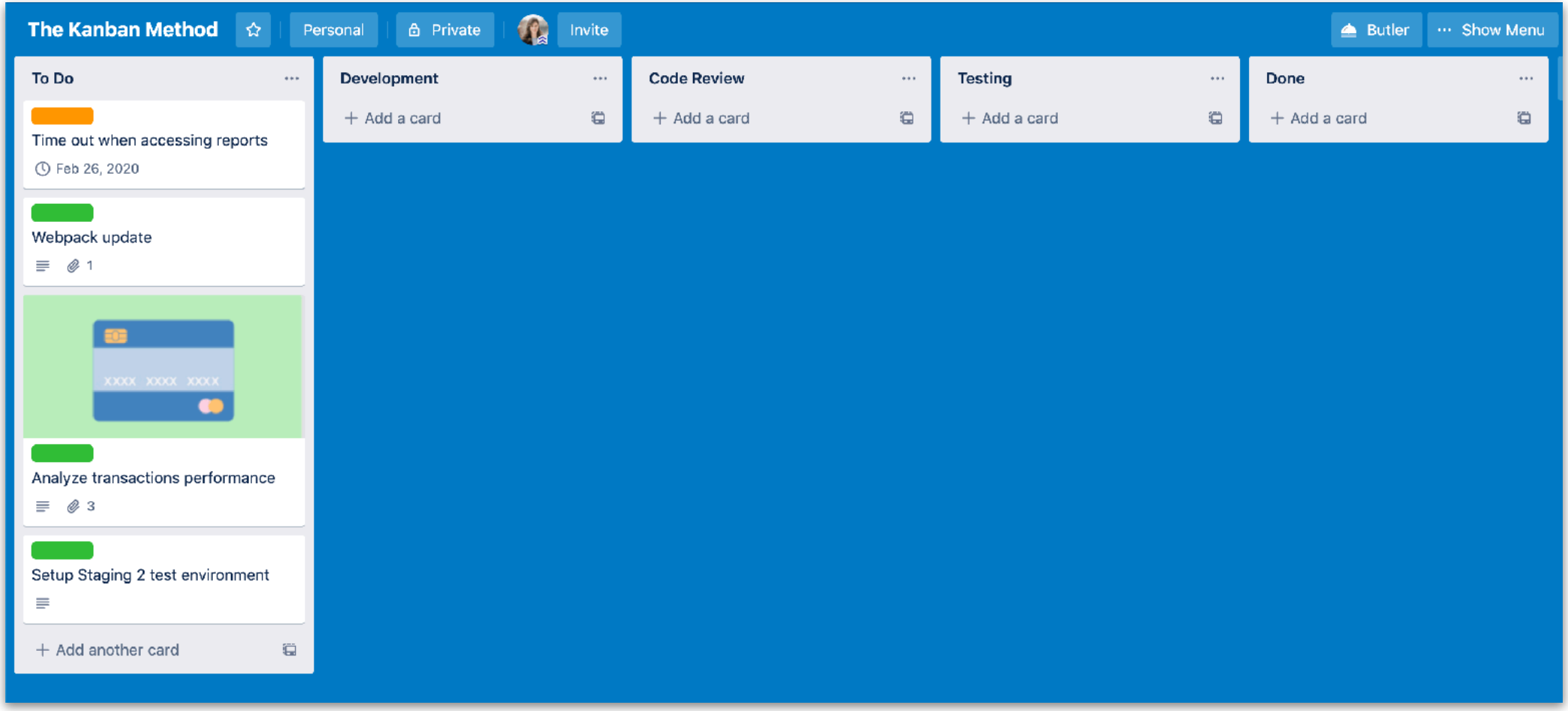


https://www.youtube.com/watch?v=iVaFVa7HYj4&t=116s

Kanban

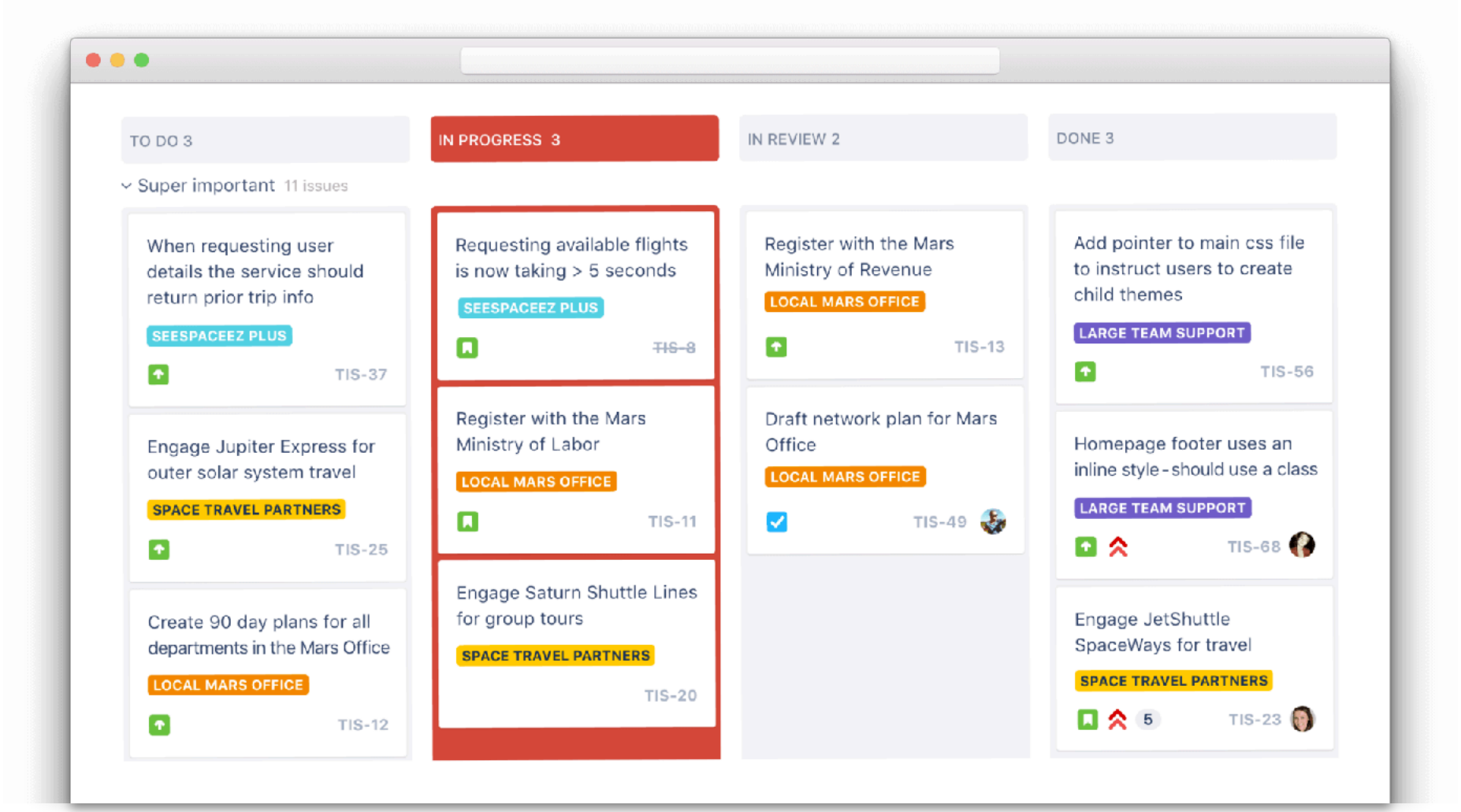
- Work-in-progress (WIP) doesn't deliver value until deployed
- Limiting / minimising WIP makes it easier to identify inefficiencies and constraints in a team's workflow
- Kanban: clearly visualise WIP and match to team's capacity
- Simple, but effective approach to help teams coordinate

Kanban – Virtual Boards



https://trello.com/en

Kanban – Virtual Boards



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

Group Exercise

Google Drive Exercise

In breakout rooms, copy and complete the template at:

http://smu.sg/XXX

Be ready to share and discuss with us!

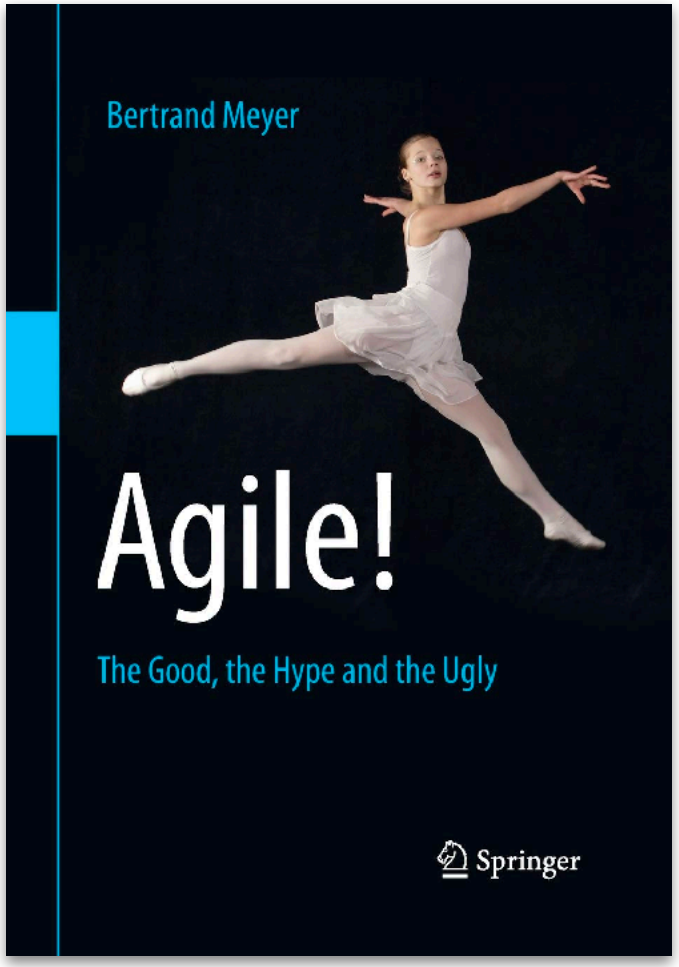
Traditional

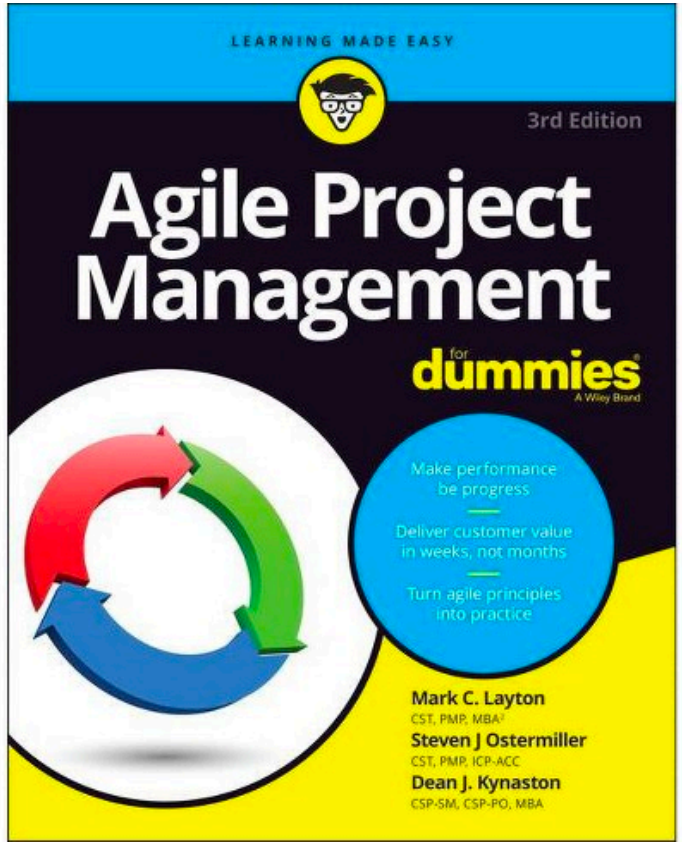
- Waterfall
- RUP
- Spiral
- Prototyping

Agile

- Extreme Programming
- Scrum
- Kanban

References / Additional Reading





(ignore the title!)