

# Xanpan

*Pronounced*

***“Zan-pan”***

*What do you get if  
you cross Kanban  
with Extreme  
Programming?*

*Team Centric  
Agile*

allan kelly

Twitter: @allankellynet - #Xanpan

<http://www.softwarestrategy.co.uk>

Mix-IT

Lyon

April 2016



# Allan Kelly...

- Consulting on software development & strategy
- Training for Agile

## Author

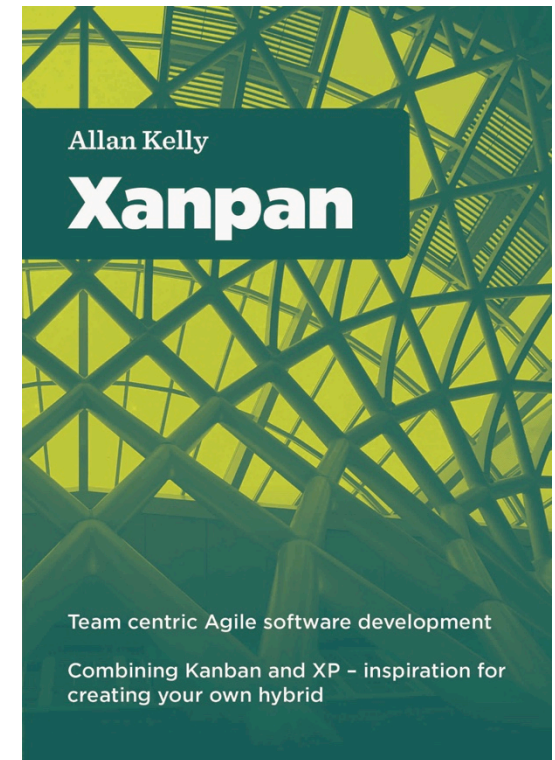
- **Xanpan**: Team Centric Agile Software Development  
<https://leanpub.com/xanpan> (2014-2015)
- **Business Patterns for Software Developers** (2012)
- **Changing Software Development: Learning to be Agile** (2008)

## Chapters in...

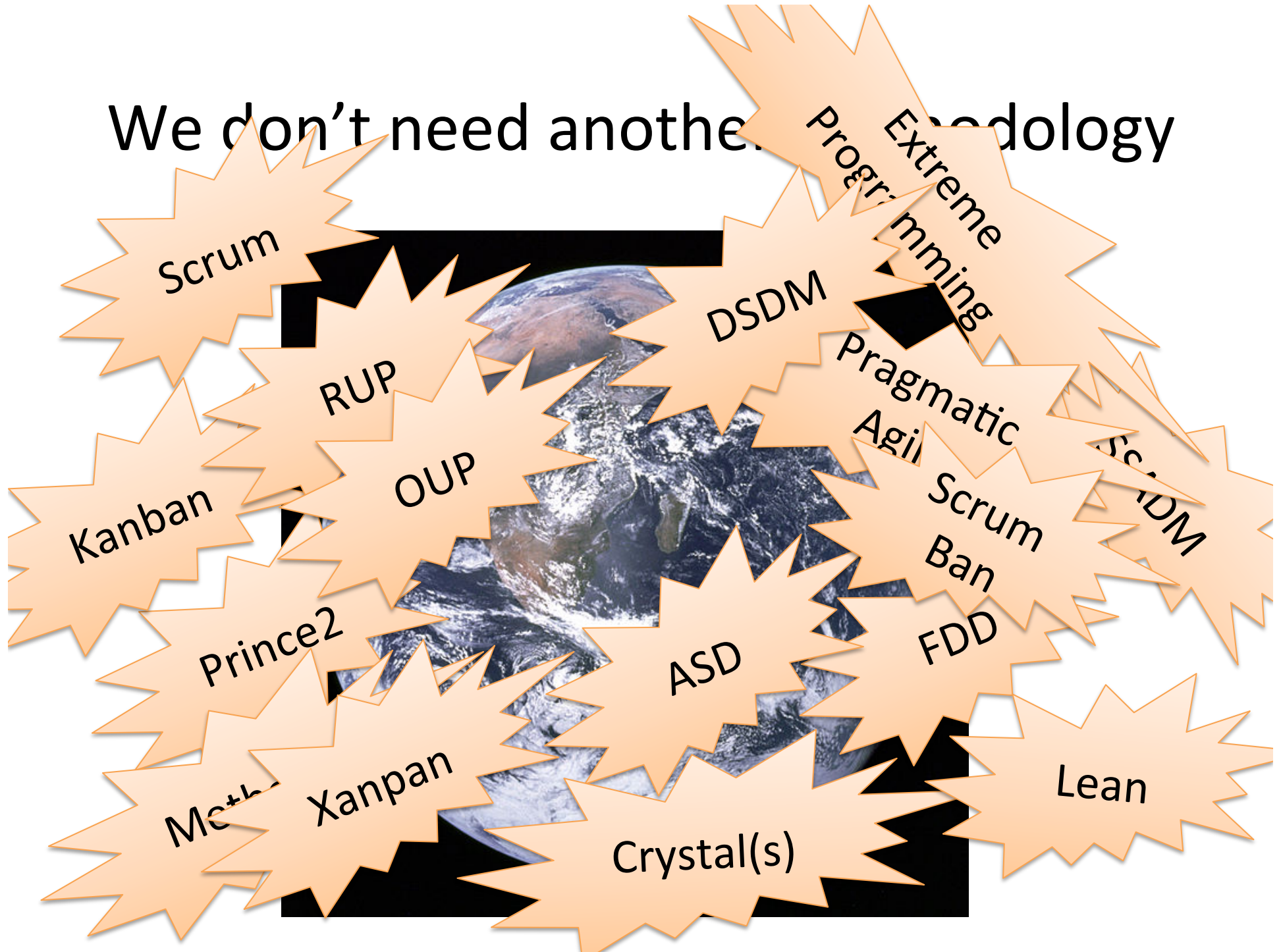
*Business Analysis and Leadership*, Pullan & Archer 2013

*97 Things Every Programmer Should Know*, Henney, 2010

*Context Encapsulation in Pattern Languages of Program Design*, vol #5, 2006



We don't need another methodology





Ken & Jeff's  
Scrum-Cola



The Real  
Thing

# Choose your Cola

8 out of 10  
teams prefer.

Will you take  
the Kanban  
challenge?



David Anderson  
Kanban-Cola



Kent Beck  
XP-Cola



High in Caffeine  
For Real Programmers

Cheap &  
Cheerful?

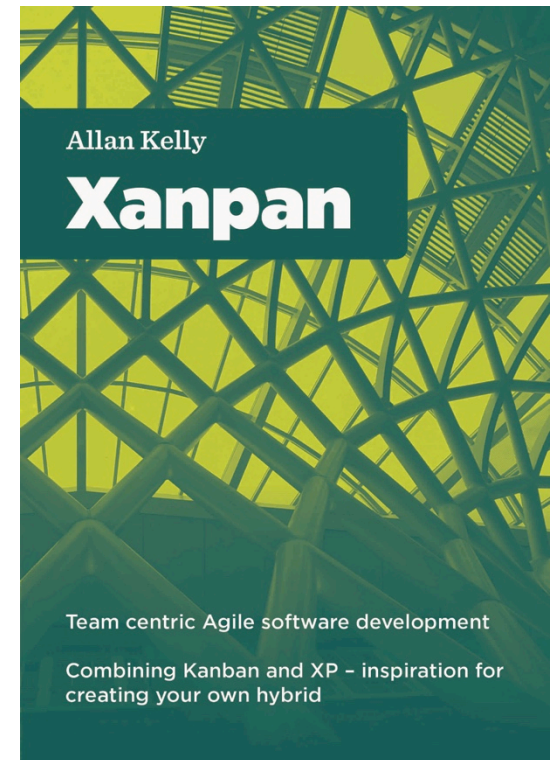


Allan Kelly - Xanpan-Cola



# Xanpan is...

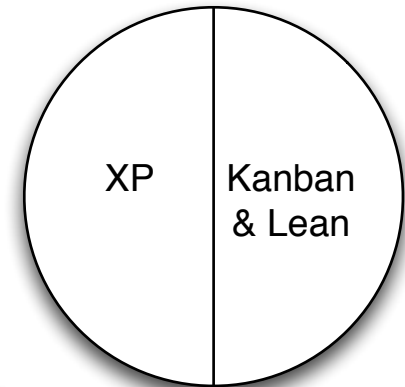
- A cross between Kanban & XP
- An example
  - Hybrid method
  - Make your own
  - Inspiration to make your own
- Team centric Agile software development
- The way Allan Kelly suggests you do things



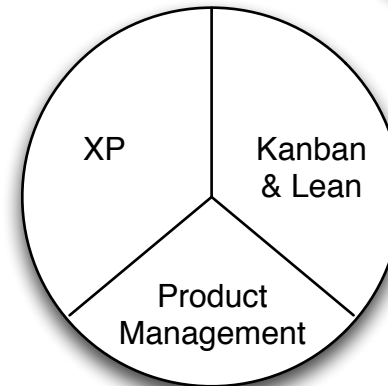
# Where did Xanpan come from?

- Experience (Lean+XP)
  - Blue-White-Red
- Kanban
- XP
- Plus
  - Seeing others
  - Reports of other cross-overs
- Making sense of what I see

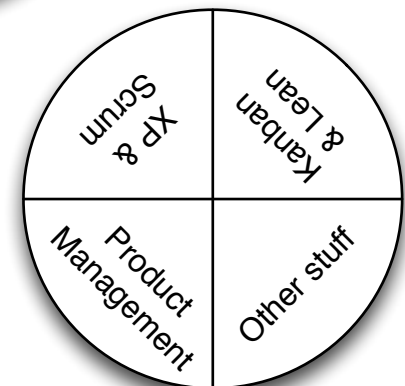
**1**  
First  
concept



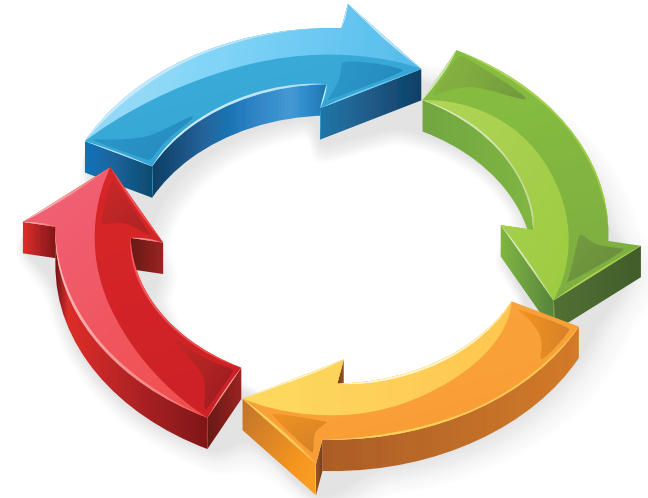
**2**



**3**



# Principles



## 1. Iteration routine

- Humans are good at deadlines

## 2. Invest in Quality - “Quality is Free”

## 3. Visualize

- *See to learn*

## 4. Dis-economies of Scale

- *Small batch size*

# Principles

## 5. Emphasize Flow

- Level, Span, Constrain

## 6. Team Centric

- Planned & Unplanned work
- #NoProjects

## 7. Goodhart's Law

## 8. Constructivism learning



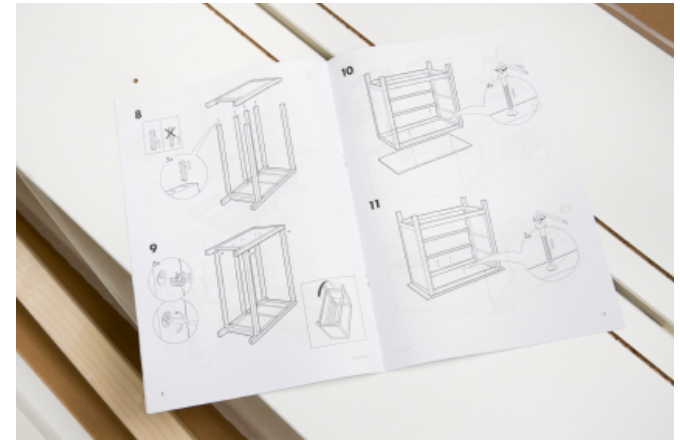


# Practice



1. XP Technical practices: TDD, CI, etc.
2. Teams can work on more than 1 stream
  - Flow multiple projects/product to 1 team
3. Break Stories to Tasks
  - Colour code work
  - Estimate in Points
  - Small is better - Think Small!
4. Benchmark against self
  - Velocity **Not** Commitment

# Practices



## 5. Flow

- Use Product “Ownership” (Product Management & Business Analysis) to restrict flow
- Apply WIP limits
- Absolute Prioritization

## 6. Planning levels (horizons)

## 7. Pick’n’Mix

## 8. Action over words

# Practices

## 9. Fit work to the time

- Deadlines are good
- Limit WIP

## 10. Evolutionary change

- *Small Bangs* are OK
- but *Big Bangs* are bad



Details.



# Team Centric



# Teams

- Keep teams together
  - Why break up successful teams?
- Flow the work to the team
- Stable teams
  - Improve performance
  - Velocity/estimation can become predictable



# Sausage Machine

Requirements & Specifications go in

Working Software Comes Out



One Team

Many projects  
Many pieces  
of work

Focus on Flow

Bring the work to the team

Bring the work to the team

Bring the work to the team

Bring the work to the team





Quality

# Quality... makes all things possible

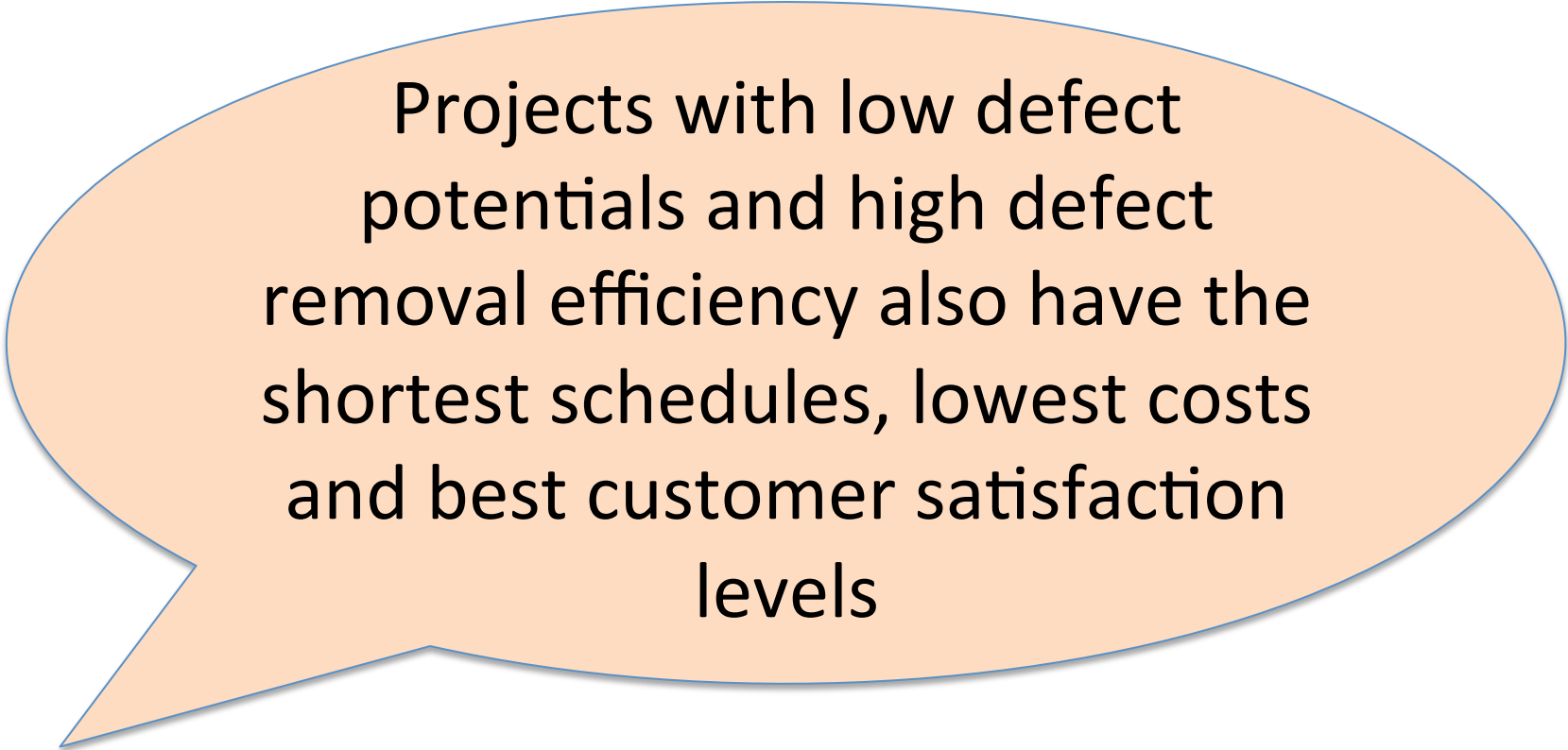
**"Quality has much in common with sex.**

- Everyone is for it. (Under certain conditions of, course.)
- Everyone feels they understand it. (Even though they wouldn't want to explain it.)
- Everyone thinks execution is only a matter of following natural inclinations. (After all, we do get along somehow.)

And, of course, **most people feel that all problems in these areas are caused by other people."**

Philip Crosby

# Quality -> Quicker

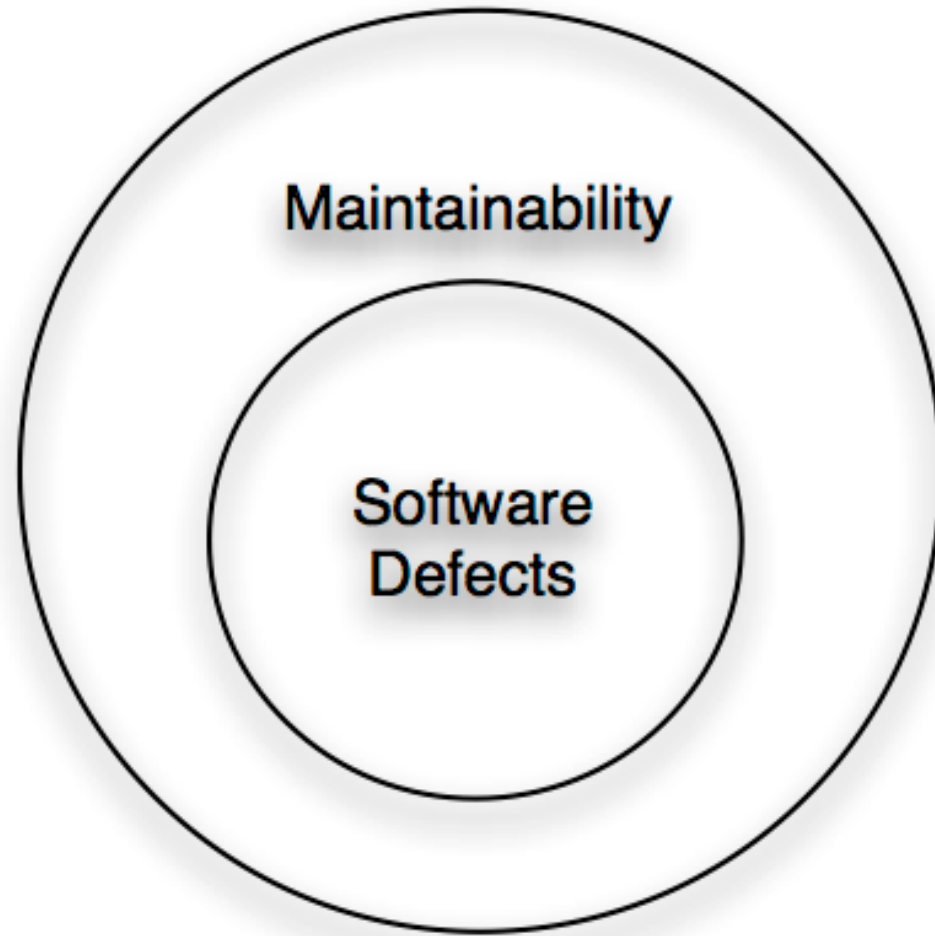


Projects with low defect potentials and high defect removal efficiency also have the shortest schedules, lowest costs and best customer satisfaction levels

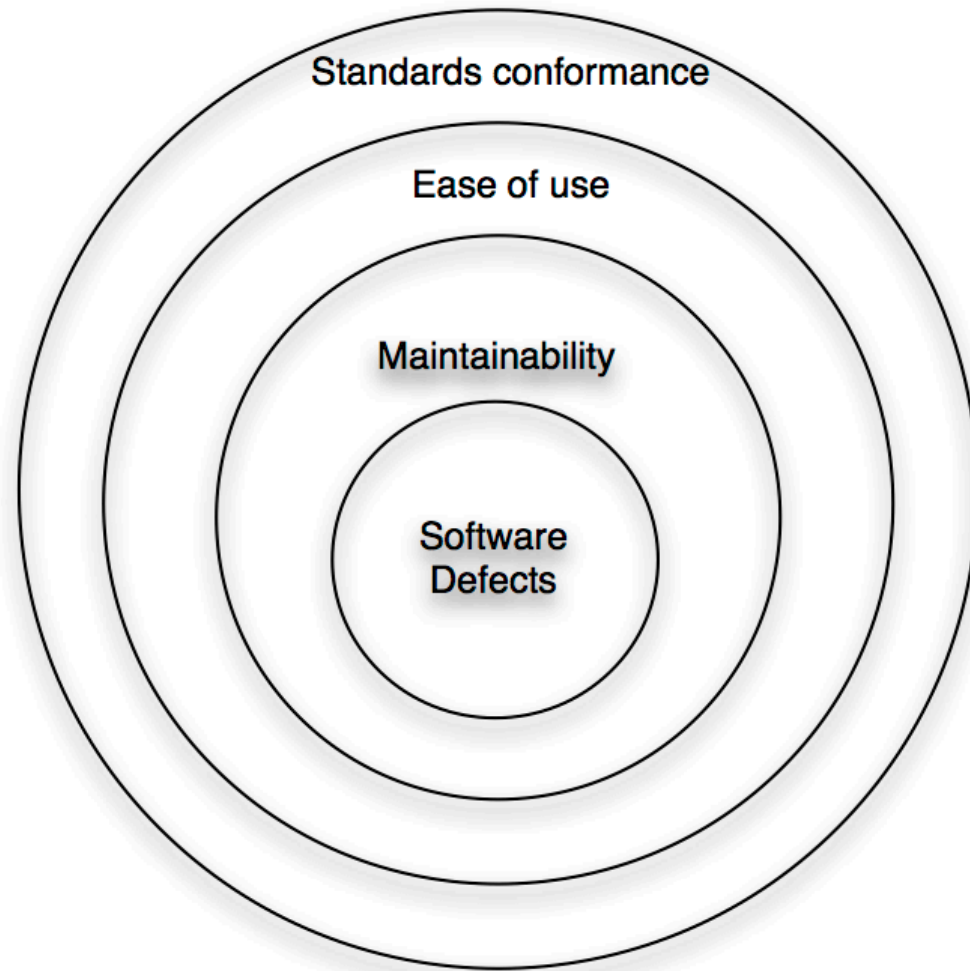
Capers Jones, 2008

*Applied Software Measurement*

# Quality core



# What qualities are important to you?



How do you improve quality?

T D D

A T D D

B D D

Visualise



**SPRINT 12**  
03/05/2011 -> 17/05/2011

**PENDING**

**CURRENT**

**REVIEW**

**FINISHED**

Printer Compatibility  
[Handwritten notes]

Printer Compatibility  
[Handwritten notes]

Printer Compatibility  
[Handwritten notes]

Printer Compatibility  
[Handwritten notes]

Printer Compatibility  
[Handwritten notes]

Printer Compatibility  
[Handwritten notes]

Printer Compatibility  
[Handwritten notes]

Printer Compatibility  
[Handwritten notes]

Printer Compatibility  
[Handwritten notes]

**BACKLOG**

[Handwritten notes]

[Handwritten notes]

Circle Summary  
[Handwritten notes]

Circle Summary  
[Handwritten notes]

Circle Summary  
[Handwritten notes]

Circle Summary  
[Handwritten notes]

Circle Summary  
[Handwritten notes]

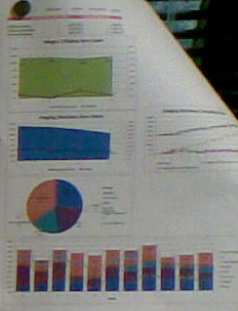
Circle Summary  
[Handwritten notes]

Circle Summary  
[Handwritten notes]

Circle Summary  
[Handwritten notes]

Circle Summary  
[Handwritten notes]

**BLOCKED**





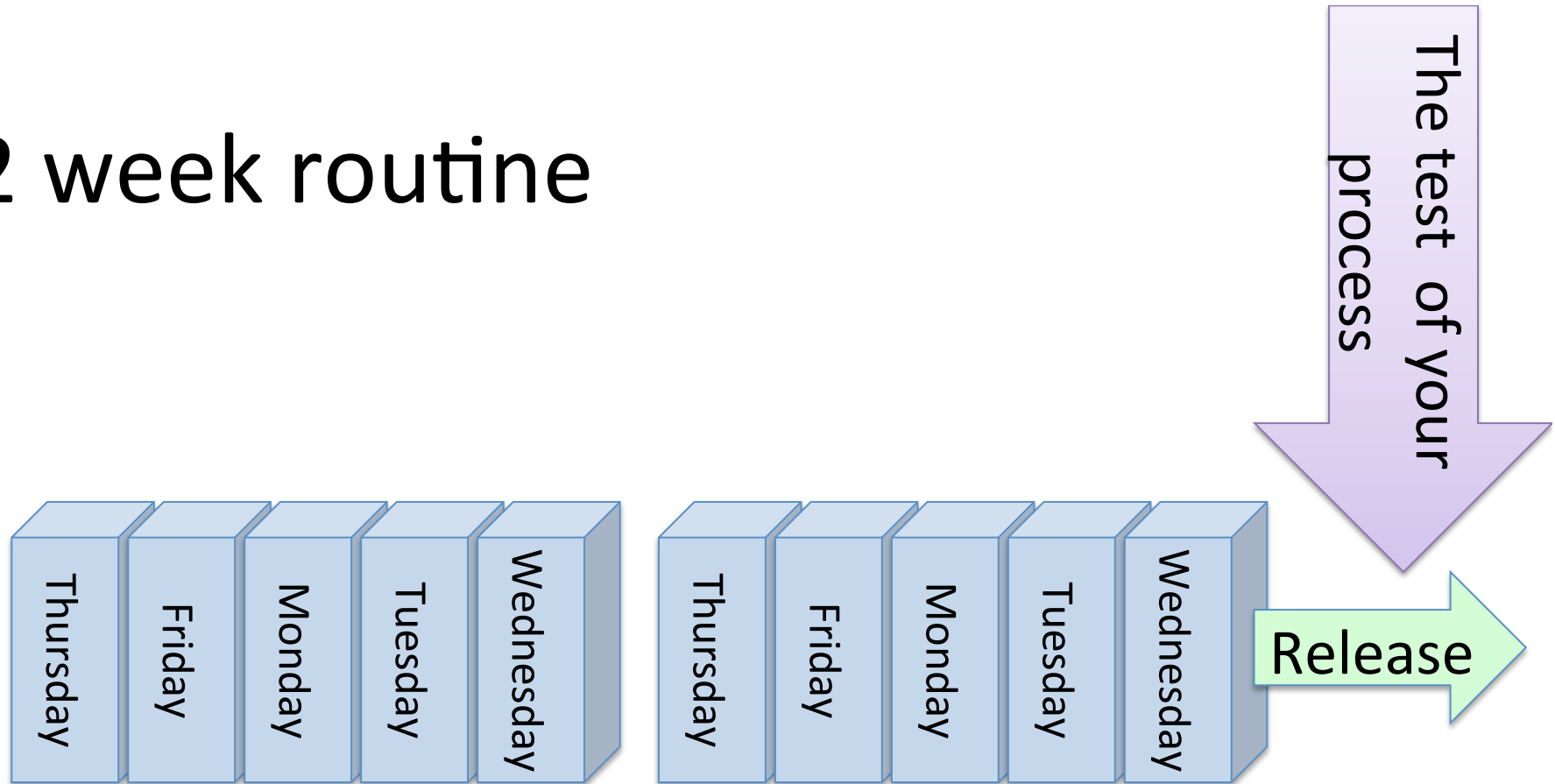
# Light Sabre

Every team  
must design  
their own  
board



# Iterations

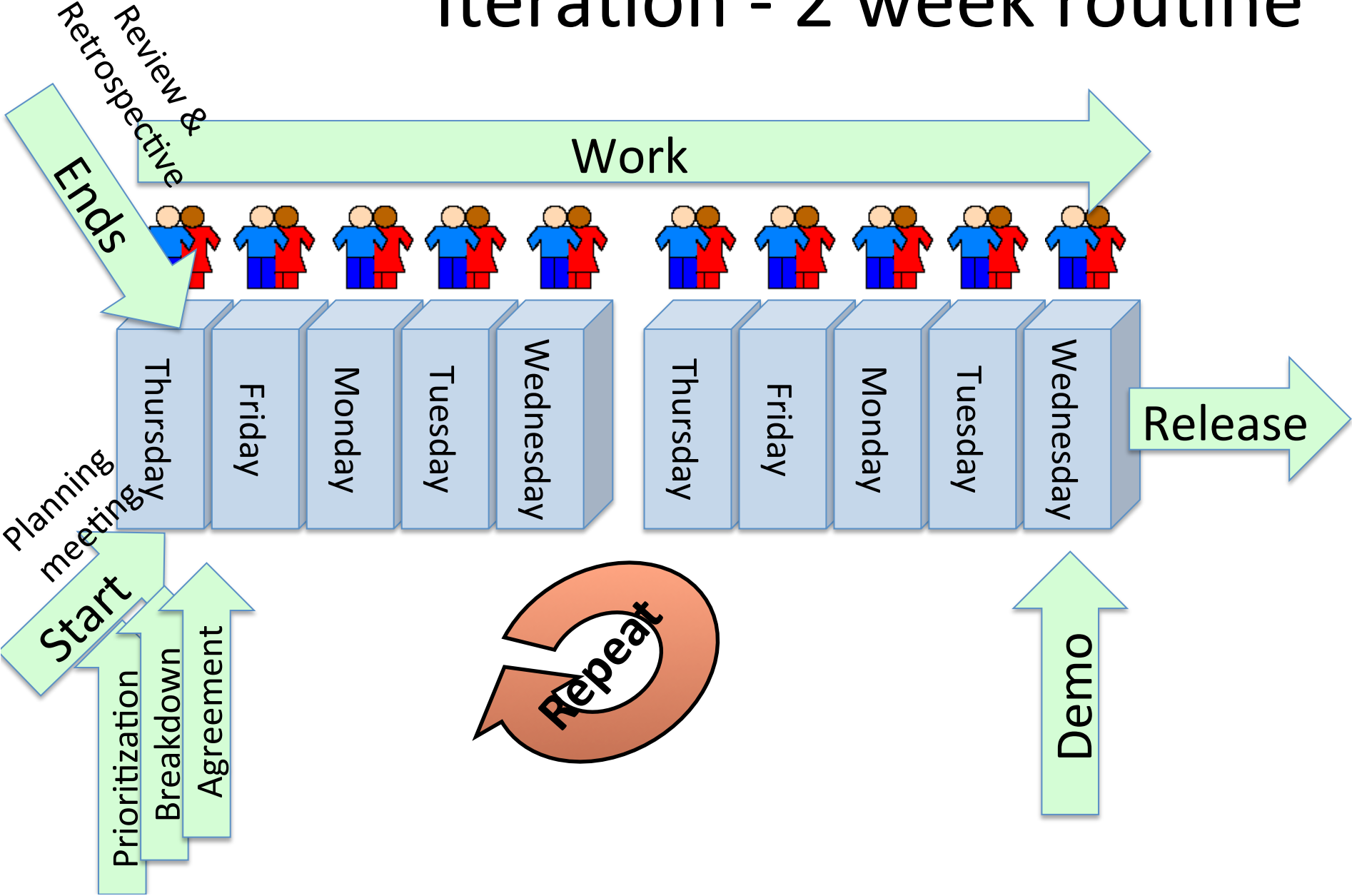
# 2 week routine



Every 2 weeks you have a shippable product

Whether you ship or not is a marketing decision

# Iteration - 2 week routine



For most teams a release every 2 weeks  
seems an impossible goal

For the best teams a release every 2 weeks  
seems an impossibly long time to wait



# Iterations & Flow

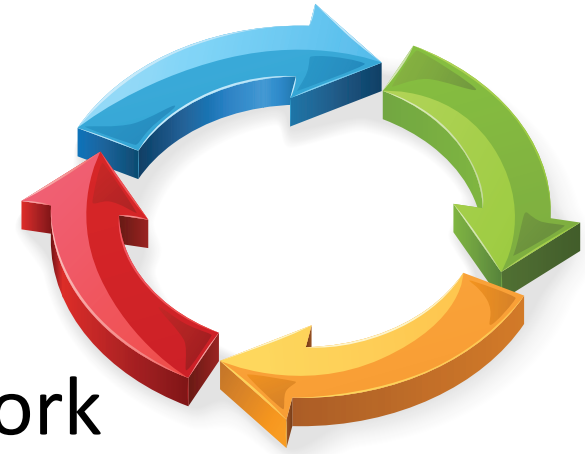
- Iterations bring structure

## **But**

- Strict iterations break flow
  - “Story must be finished in sprint”
  - “Story cannot be bigger than a sprint”
  - Sprint tail overwhelmed by finished stories
  - Testers drop standards
- Strict iteration
  - Difficult at first – learn to think small



# Iterations & Flow



- Stories spanning sprints levels work
  - Break down stories to tasks
  - Tasks only counted when completed
  - When all tasks done, Story done
- *3 Strikes and you are out!*
  - Story span 1 Iteration, OK, it happens
  - Story spans 2 Iterations, umm... Red Flag
  - Story spans 3 Iterations, Out! Story too big



# Deadline are good

- Humans are
  - Very bad at estimating time
  - Very good at meeting deadlines
- So harness deadlines
  - 2-week iteration deadlines
  - Work to the deadline
  - Synchronize on deadlines
  - Flex the work within the deadline



# Unplanned work allowed

- Seek value
- Reflect reality
- Nothing wrong with late work
  - Just because work arrives late does not mean it is less valuable
  - Late breaking work may be more valuable



# Planned & Unplanned work



- Work planned in planning meeting
- Unplanned work allowed at any time
  - Tag it, e.g. Yellow card
  - Retrospective estimation
- At end of the iteration count points unplanned
  - Graph/Track planned v. unplanned
  - Incorporate into planning velocity



# Breakdown

- In planning meeting
- Part
  - Software Design
  - Requirements elicitation
  - Opportunity to reduce scope
  - Estimation exercise



Image from Paul Goyette, Creative Commons License  
[http://commons.wikimedia.org/wiki/File:Wrecking\\_ball.jpg](http://commons.wikimedia.org/wiki/File:Wrecking_ball.jpg)

# Stories: 2 Golden Rules

1

**As a**  
**I want to**  
**So that**

*Role or Persona*  
*Do a Something*  
*Objective*

2

Story should benefit business  
(Story should have value \$s & €s)

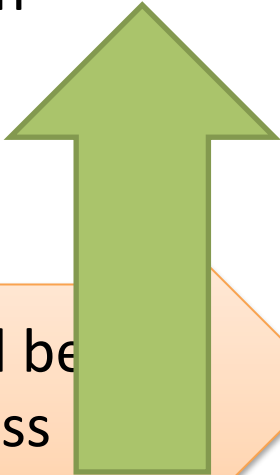
**Bang!**

Story should be small –  
deliverable in days; max 2  
weeks

Value but too big to deliver soon

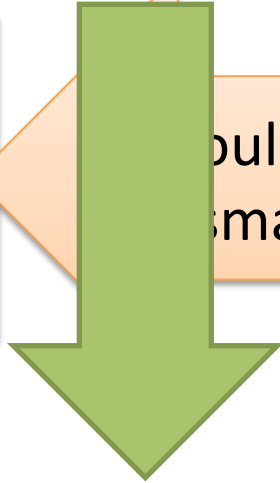
**EPIC**

Should be business



**As a** *Role or Persona*  
**I want to** *Do a Something*  
**So that** *Objective*

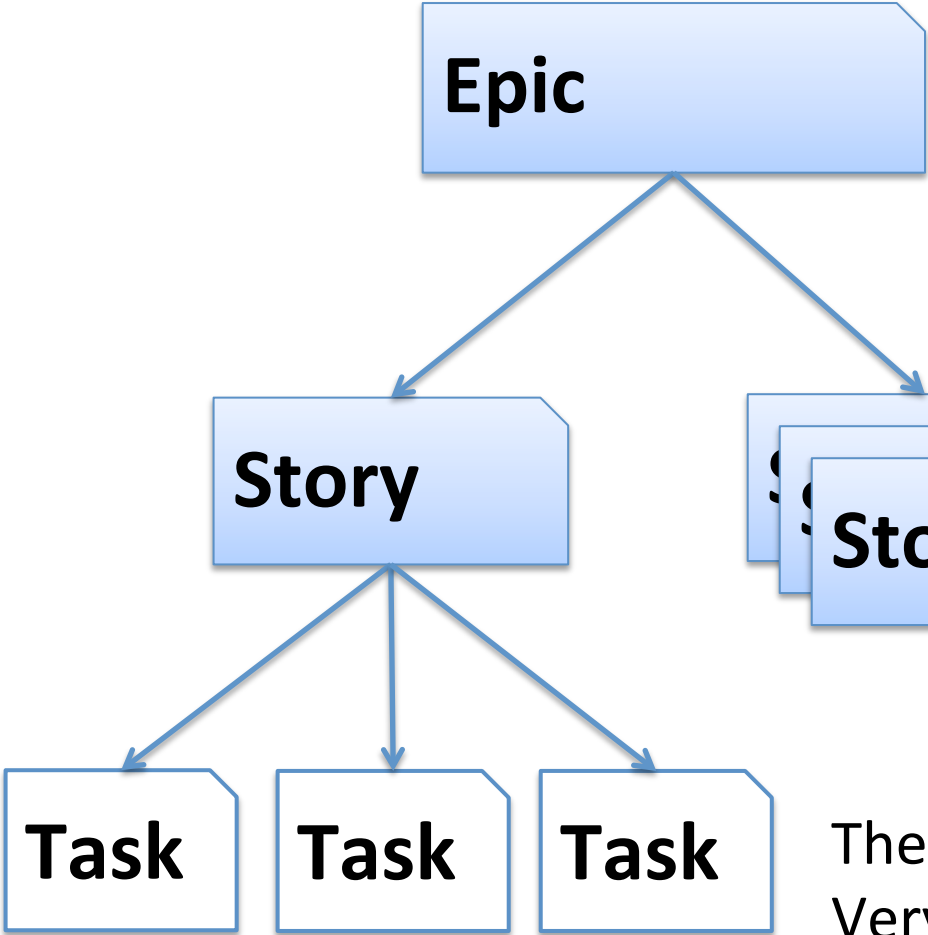
Should be small



**TASK**

Small enough to deliver really soon but lack business value

# Epic-> Story-> Task



Very valuable but BIG  
Used for forward planning  
Breakdown to stories as they approach

Stories have value  
Small enough to delivery soon  
Have acceptance criteria

The things you do to build a story  
Very small but no business benefit

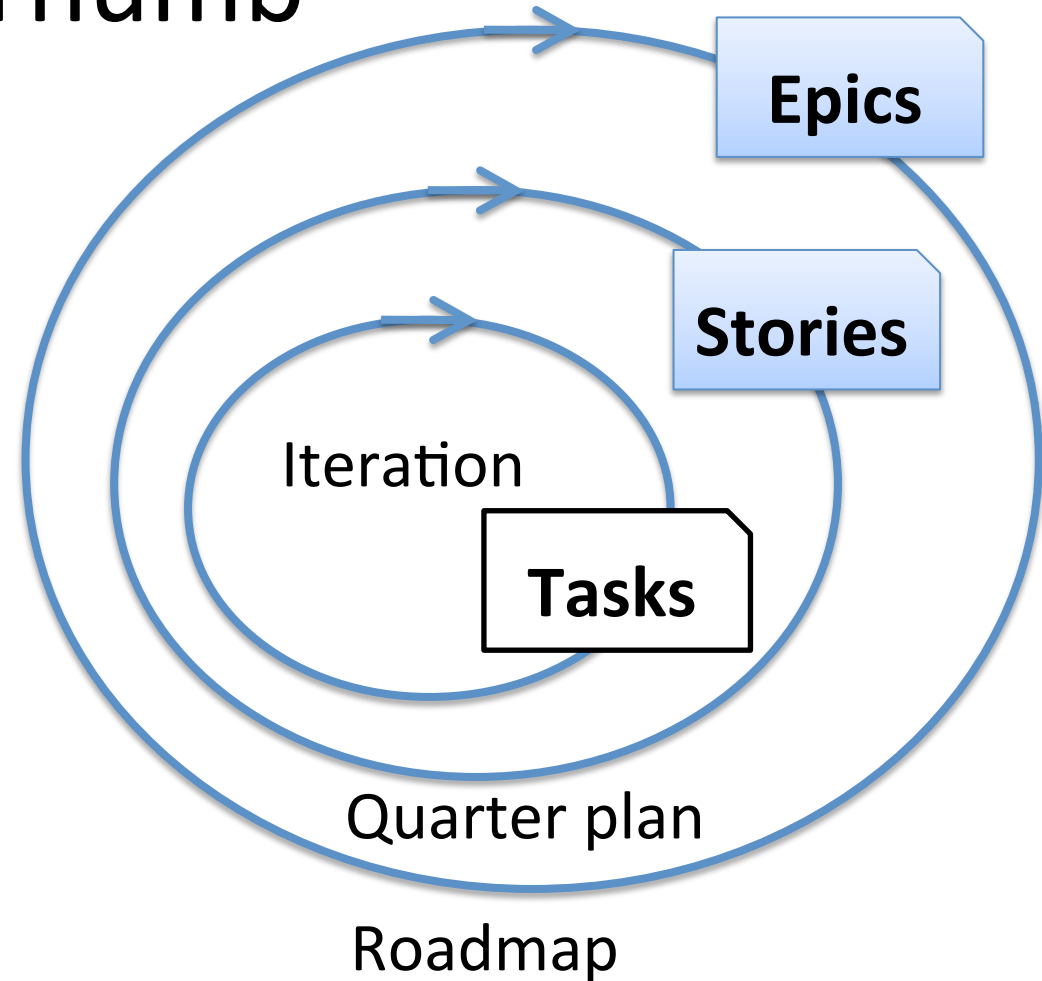


# Rule of Thumb

Iteration plan with  
Task level

Quarter / Release  
plan with Stories

Roadmap plan with  
Epics





# Focus on Value not The End

Ask not, “When will the software be done?”

But ask: “When will the software deliver value next?”



Think: Stream of Value

(which might stop one day)

Not: An end date

**Reds**



**Yellows**

Unplanned work

**Green**

Specific to you

# Goodhart's Law

Any observed statistical regularity will tend to collapse once pressure is placed upon it for control purposes.



Velocity & points break down if abused...

... and so do other measurements

# Scaling in 1 slide

Streams not projects (#NoProjects)

Teams over projects

Decouple teams

Independent teams

Technical practices to bind them

Common reporting not common working

# Is Xanpan useful?

- Maybe
  - Take it
  - Use it
- Inspiration
  - Roll your own



Image from Ildar Sagdejev under Creative Commons license  
[http://commons.wikimedia.org/wiki/File:2009-02-15\\_Rolling\\_a\\_cigarette.jpg](http://commons.wikimedia.org/wiki/File:2009-02-15_Rolling_a_cigarette.jpg)

# Decide for yourself

## EBook

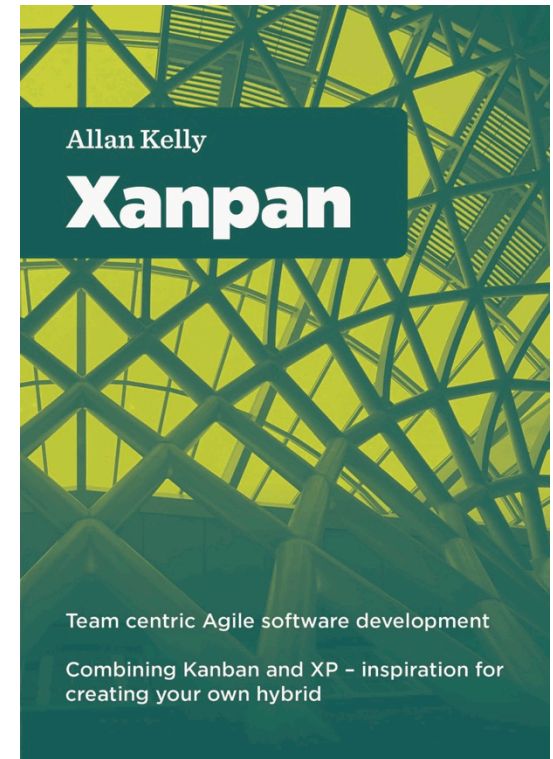
Half price, \$4 until 31 April 2016

<http://leanpub.com/xanpan/c/MixIT2016>

Discount code: **MixIT2016**

**Printed** from Lulu.com

<http://tinyurl.com/zf5hke4>



*Which brand of Cola  
are you drinking?*



allan kelly

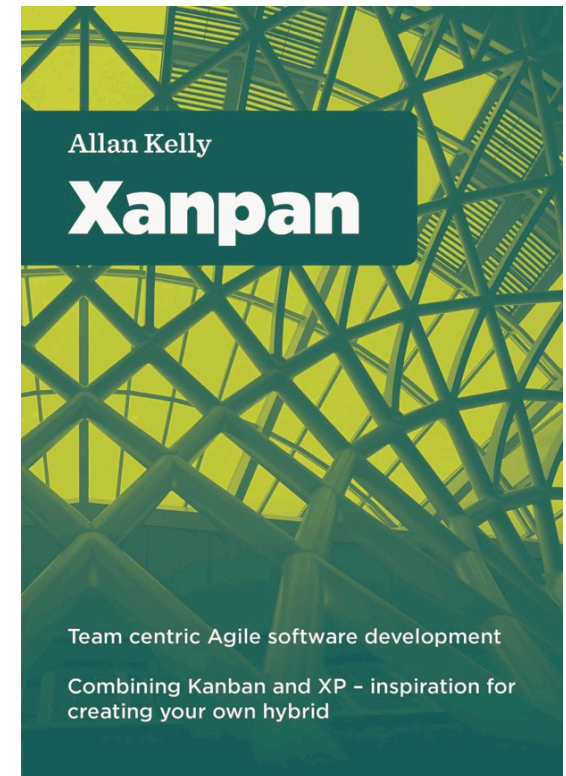
[www.allankelly.net](http://www.allankelly.net)

[allan@softwarestrategy.co.uk](mailto:allan@softwarestrategy.co.uk)

Twitter: [@allankellynet](https://twitter.com/allankellynet)

<http://leanpub.com/xanpan>

Discount code: **MixIT2016**



# Appendix





Look beyond the  
label

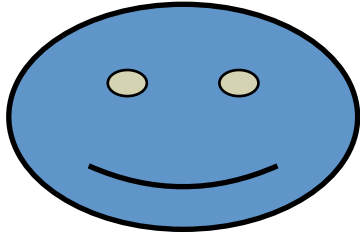
Waterfall

Agile



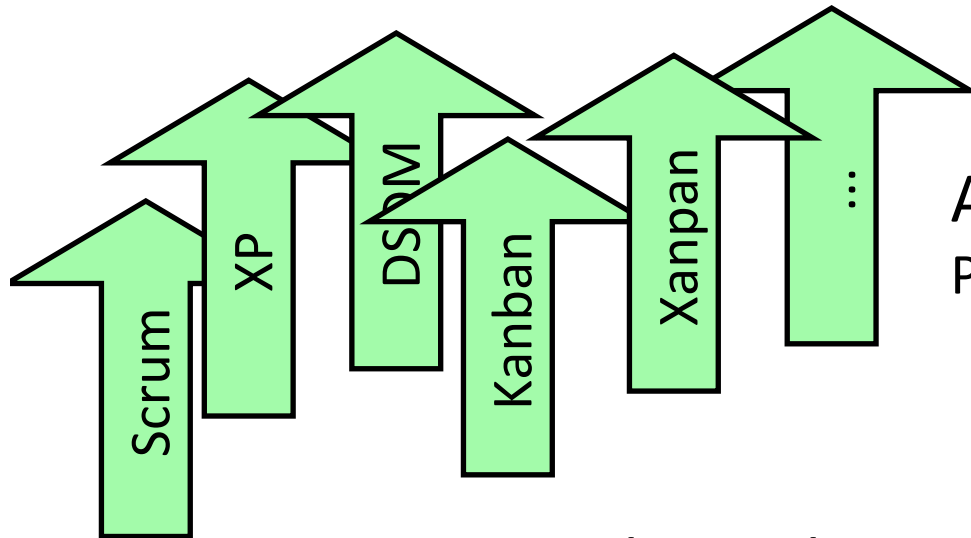
Jonathon's Run Fall, Pennsylvania by Hubert Stoffels (  
<http://flickr.com/photos/22195940@N00>)  
Creative Commons License

# Agile, Agile methods & the Agile toolkit



The State of Agile (our objective)

- Quick on our feet
- Respond to change rapidly
- Deliver quickly



Agile Methods

Promise to create the state of Agile



The Agile toolkit

- Test Driven Development, Refactoring
- Iterations, Time boxing
- Retrospectives, ....

# Bigger Teams Better

- Less susceptible to variability
- Losing a member isn't so bad
- Growing team more efficient
- More predictability
- Easier to staff with all the skills
- 4 to 15 people – Everyone!
  - Coders, Tester, Analysts (PO) and anyone else needed!



# Yes, Estimation



- Estimate White tasks in planning meeting
  - Ball-park estimate Blues
- Estimates in Points
  - Your currency £ \$ €
  - One currency
  - Forget hours
- Estimation helps design thinking

I've come to like Planning Poker but choose your own poison

# Estimates are for the team

- How much work to put in next iteration
  - Are we taking on too much?
- Assist breakdown
  - Breakdown is design
- Estimates allow for red-flags



# Estimation worthwhile?

“I can bring a project in to the day”

- For scheduling? Perhaps
    - Some teams report good results
    - Some teams placebo effect
    - Long run average accurate enough
  - Provides Developers with safety valve
  - Useful input to design process
- (Forget *actuals* – retrospective estimates)



# Estimation...

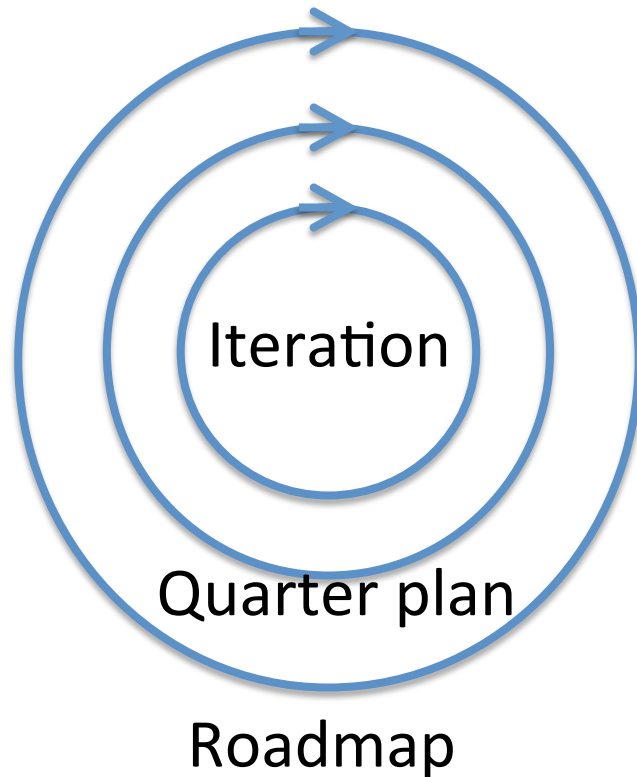


- For work within 3 months can be
  - Generally right
  - Useful in designing & scheduling
- Effort estimates beyond 3 months too variable
- Value estimate beyond 3 months essential

## Estimate value before effort

- Close the loop & evaluate afterwards

# 3 Planning Horizons



## Iteration (Sprint)

- 2-4 weeks ahead

## Quarter plan (Release)

- Next quarter
- 2-4 releases ahead
- (2-8 Iteration)

## Roadmap

- 1-2 years by quarter
- 2-5 year ahead



# Scaling...

How do you scale Xanpan?



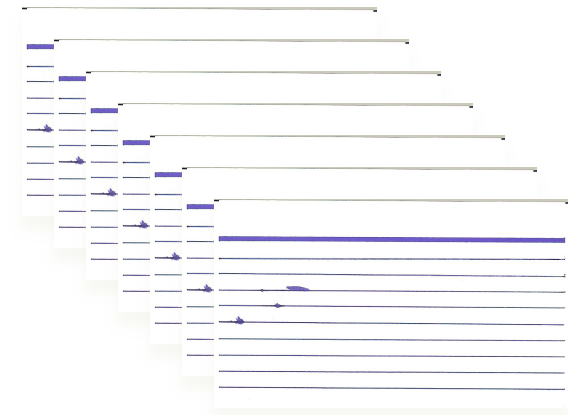
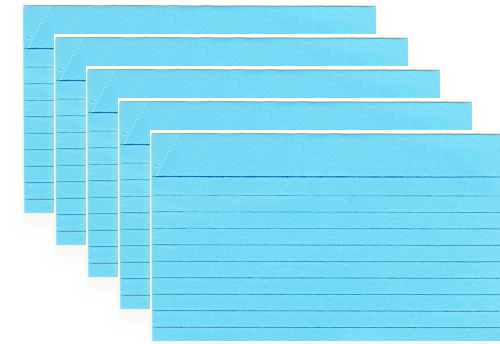
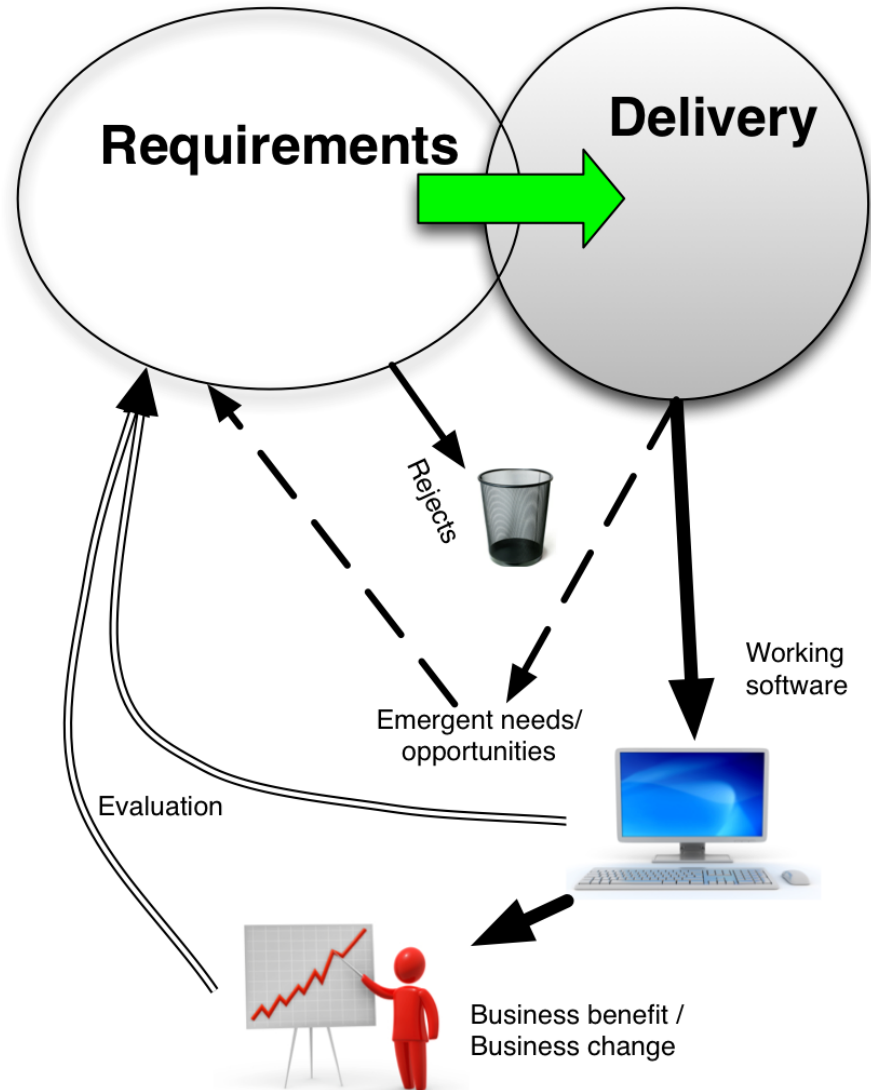
Do you mean...

- How do I manage a large team?
- How do I manage multiple teams?
- How do I govern Agile working?

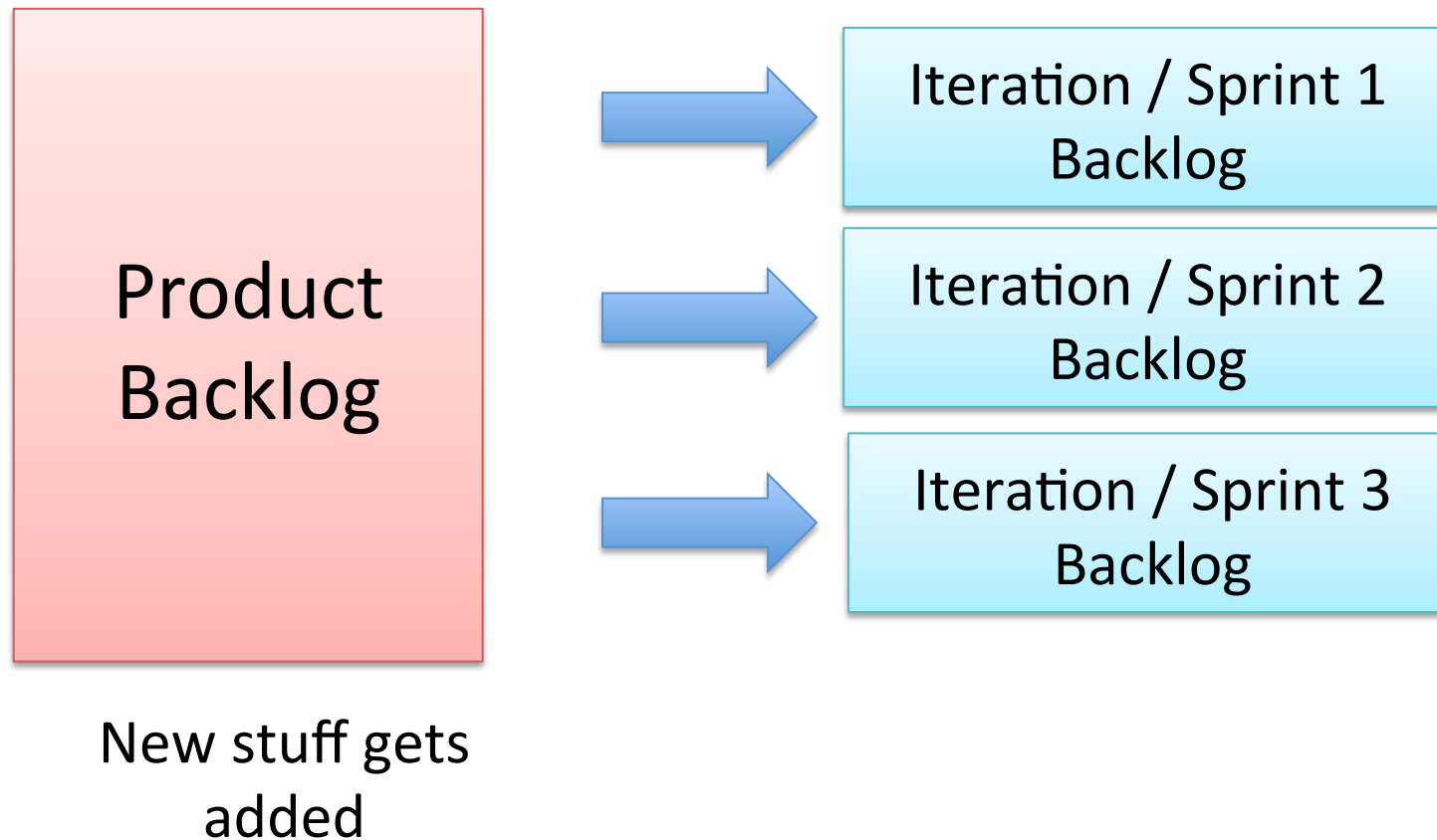
Each question has a different answer, lets do coffee ☺



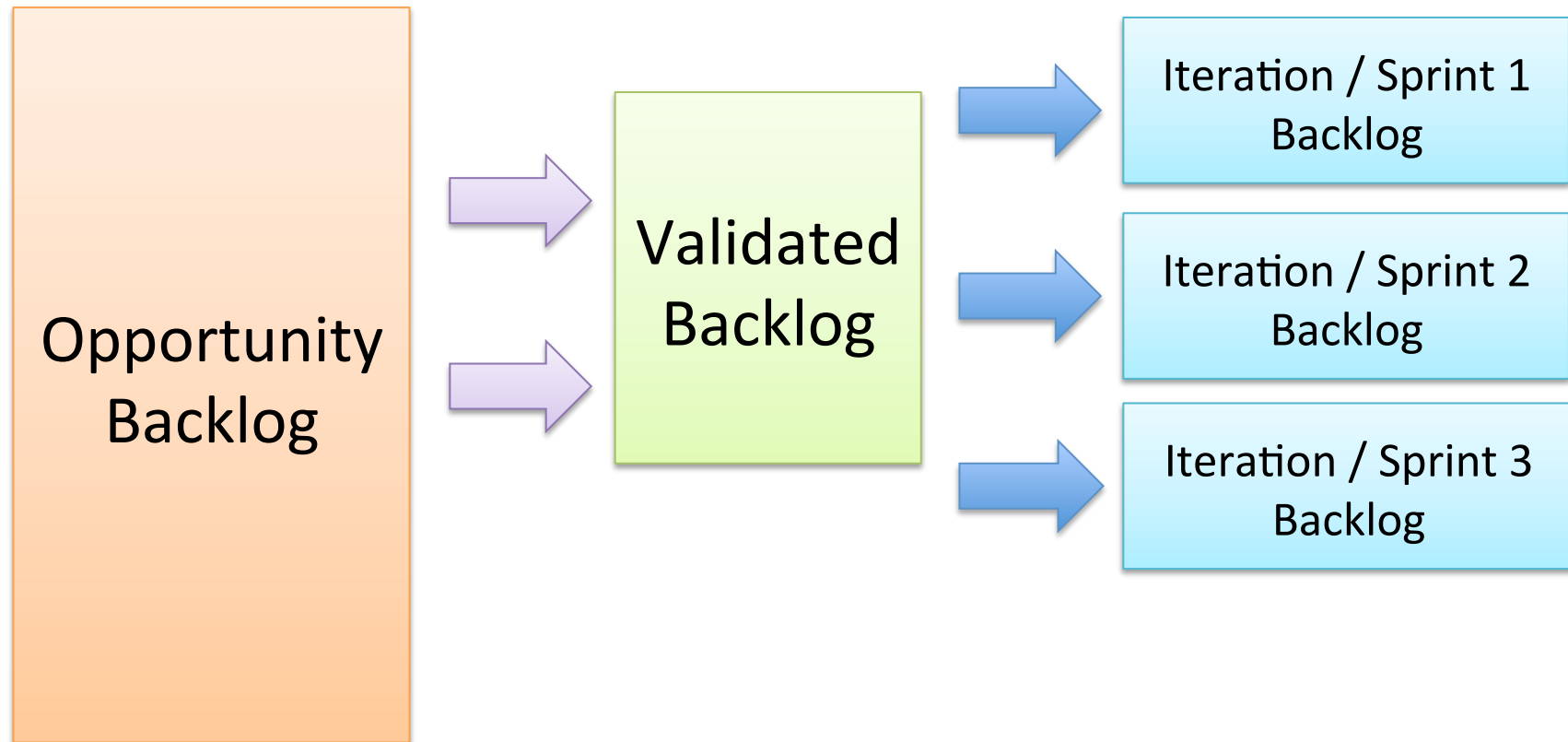
# Three backlogs



# Classic Scrum



# 3 Backlogs recommended



# Opportunity backlog

- Can grow as big as you like
- Never pretend it will all be done
- Continually
  - Add new ideas
  - Evaluate existing ideas
  - Trash some
  - Move some to validated backlog

# Validated backlog



- Much smaller than opportunity
  - Limited to N future iterations capacity
  - $1 < N < 6$  iterations (@ 2 weeks = 12 weeks)
- Stuff that has been validated
  - Has provable business value
  - Among highest value
- Will be worked on soon
- Review / repopulate every iteration

# Validated backlog is Quarter plan

Iteration					
+1	+2	+3	+4	+5	+6
					

# Validated backlog



- All cards assigned VALUE on entry to validated backlog
- Ball-park estimates added if needed
- Ball-park estimate & current velocity gives planning horizon
- **NOTHING IS CERTAIN**