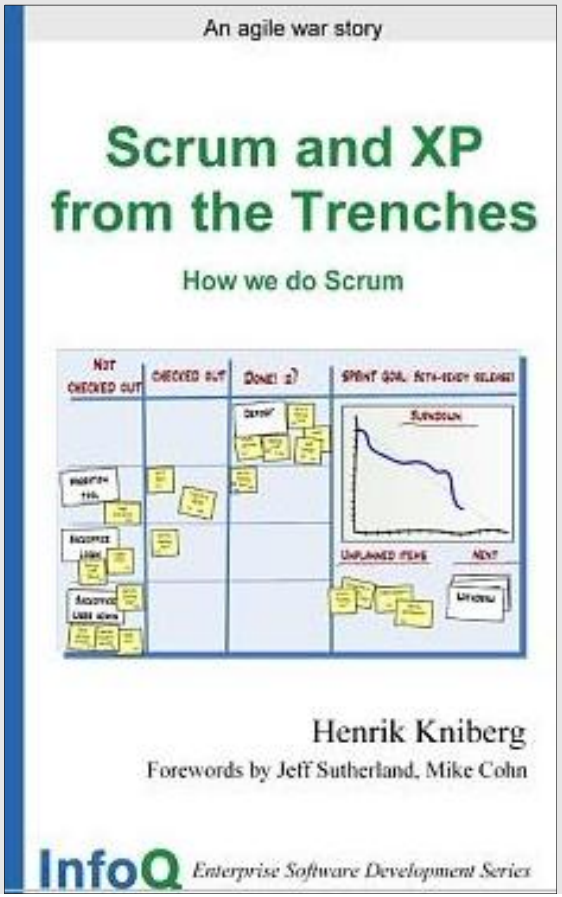
Kanban

A Lean approach to Agile software development

JFokus
January 26, 2010



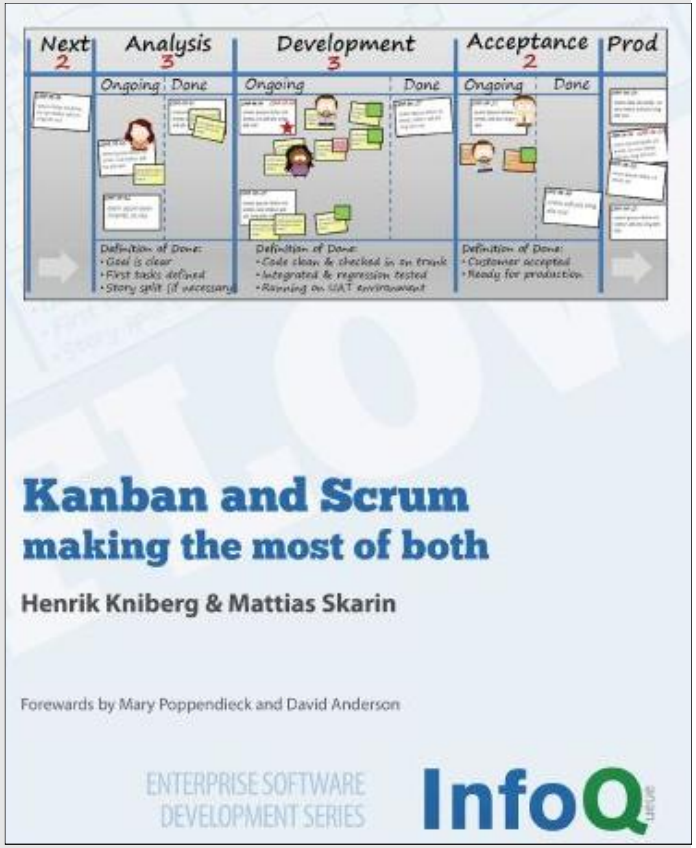
Henrik Kniberg
Agile/Lean coach
www.crisp.se

Agile Alliance Board of directors

CERTIFIED Scrum Trainer

henrik.kniberg@crisp.se
070 4925284

crisp



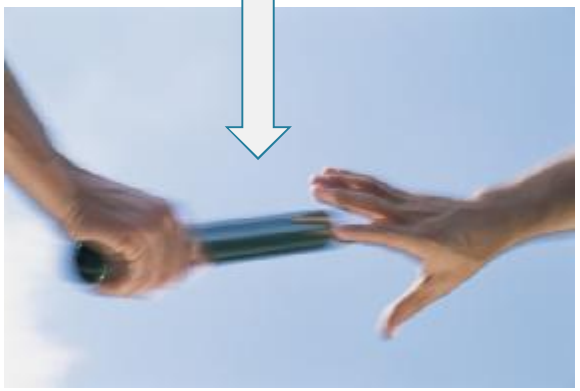
Goals of this tutorial

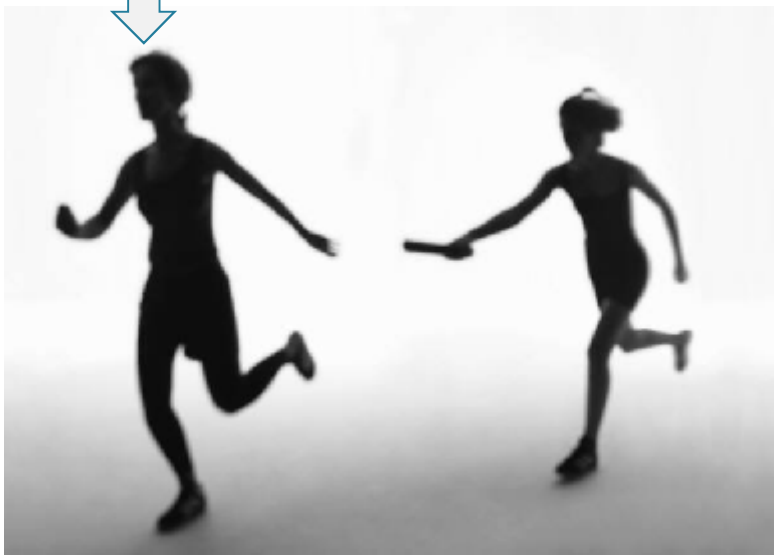
- **Basic understanding of Kanban**
  - Where it came from
  - What problem it tries to solve
  - Underlying Lean principles
  - How it looks like in practice
  - How to get started

Exercise

Flow

Watch the baton, not the runner!





Cycle time

Cycle time / lead time
= The reliable, repeatable time from customer need until that need is satisfied (≈ time to market)

Examples:
- Time to get my hamburger
- Time to resolve support request
- Time to get home from work
- Time to deliver a feature

Not to be confused with Iteration/Sprint length
= release frequency / cadence length

Minimize cycle time

Watch the baton!

- Competitive advantage
- Detect defects earlier
- Less time for change to happen

Problem: Invisible baton

Team 1

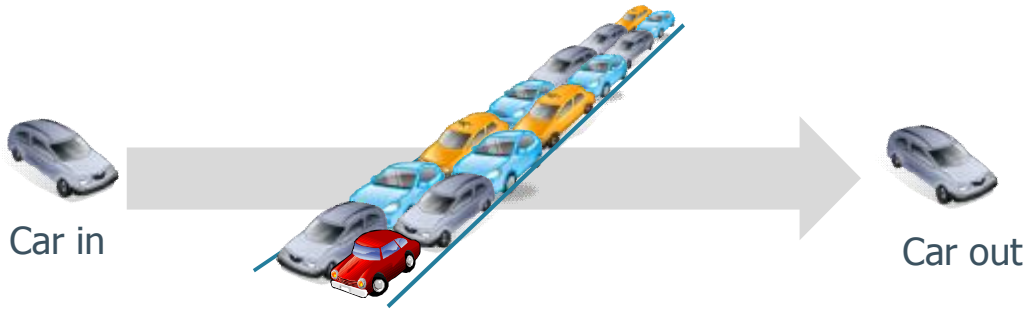
Team 2

Managers who don't know how to
measure what they want
settle for
wanting what they can measure

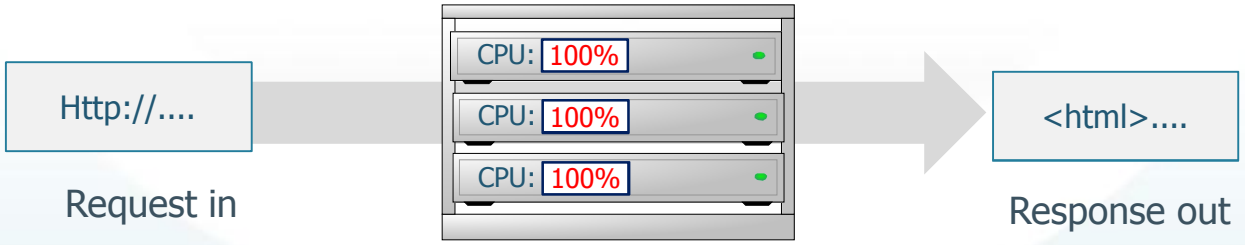
Russel Ackoff

Optimizing resource utilization = suboptimizing cycle time

100% runner utilization
=> Lose the race



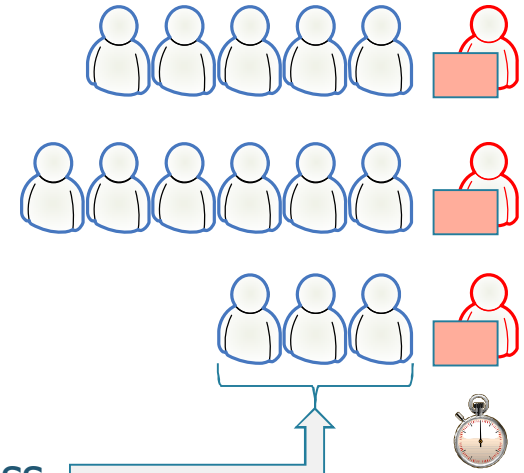
100% Road utilization
=> Long time to get home



100% Server utilization
=> Long response time

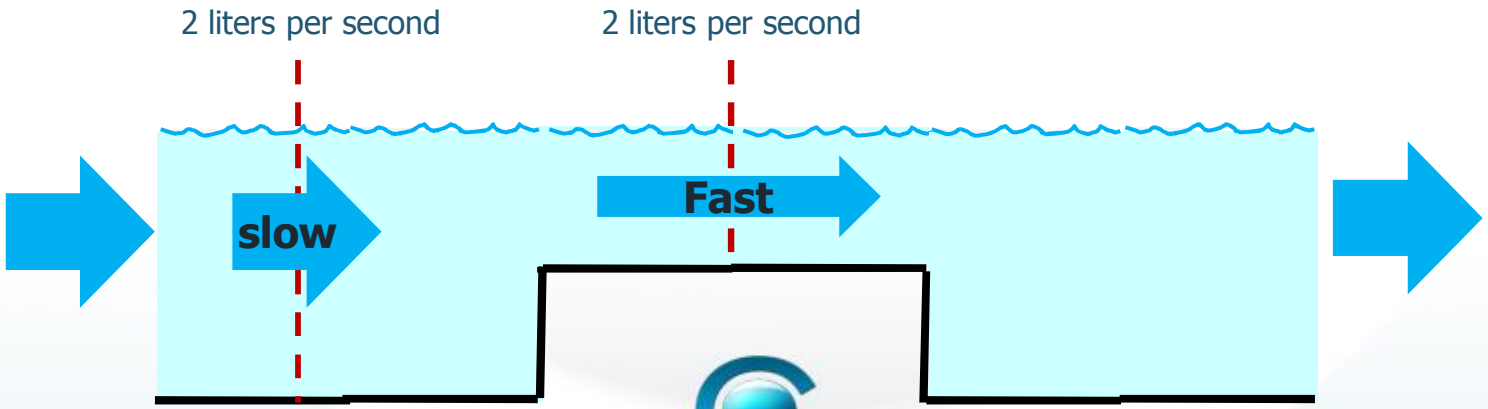
Queuing theory: Little's law

Smaller batch size => faster cycle time



Little's law

$$\text{Total Cycle Time} = \frac{\text{Number of Things in Process}}{\text{Average completion rate}}$$



Minimize batch size
Knowledge is perishable

1 fresh banana per day

Monday	Tuesday	Wednesday	Thursday	Friday
				

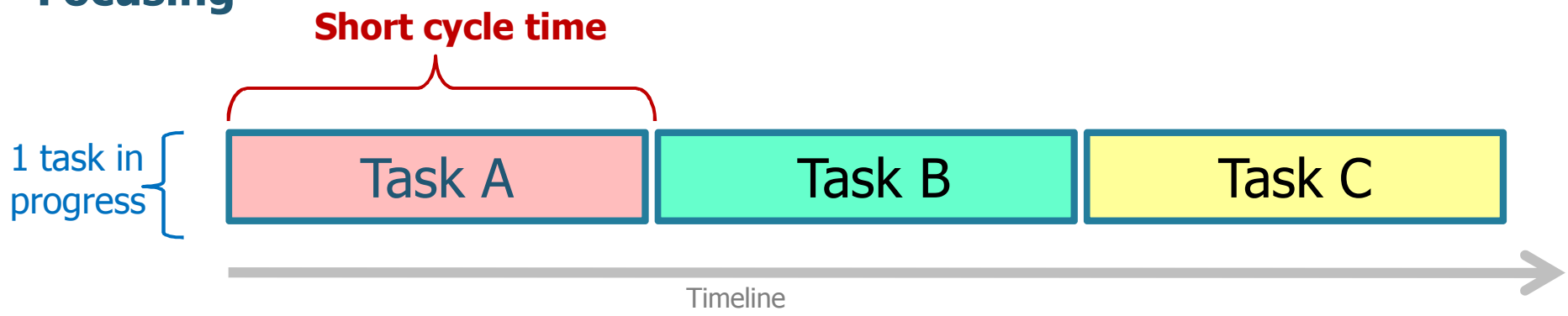
... or 5 rotten bananas per week?

Monday	Tuesday	Wednesday	Thursday	Friday
				

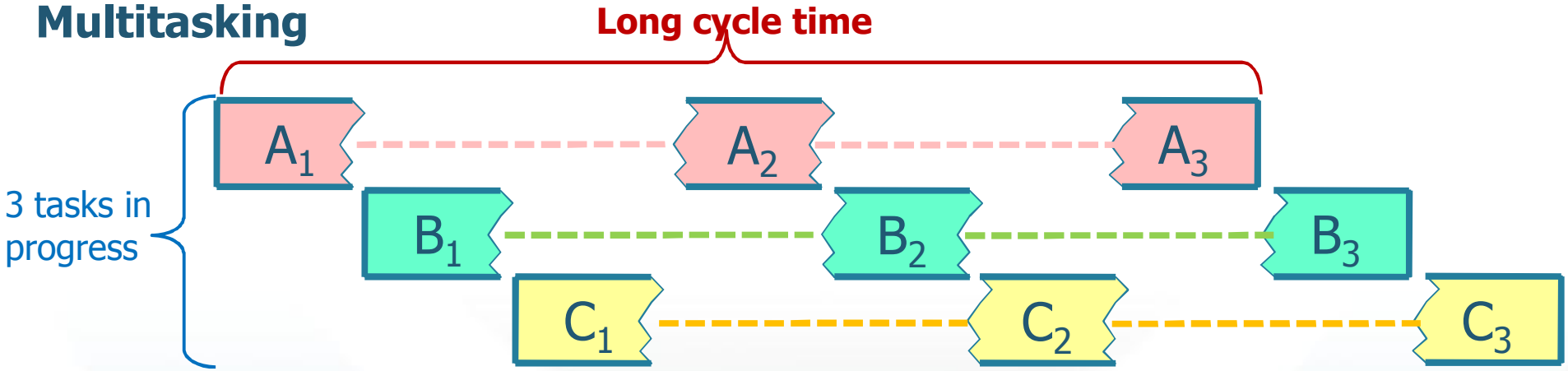
Little's law in action

$$\text{Total Cycle Time} = \frac{\text{Number of Things in Process}}{\text{Average completion rate}}$$

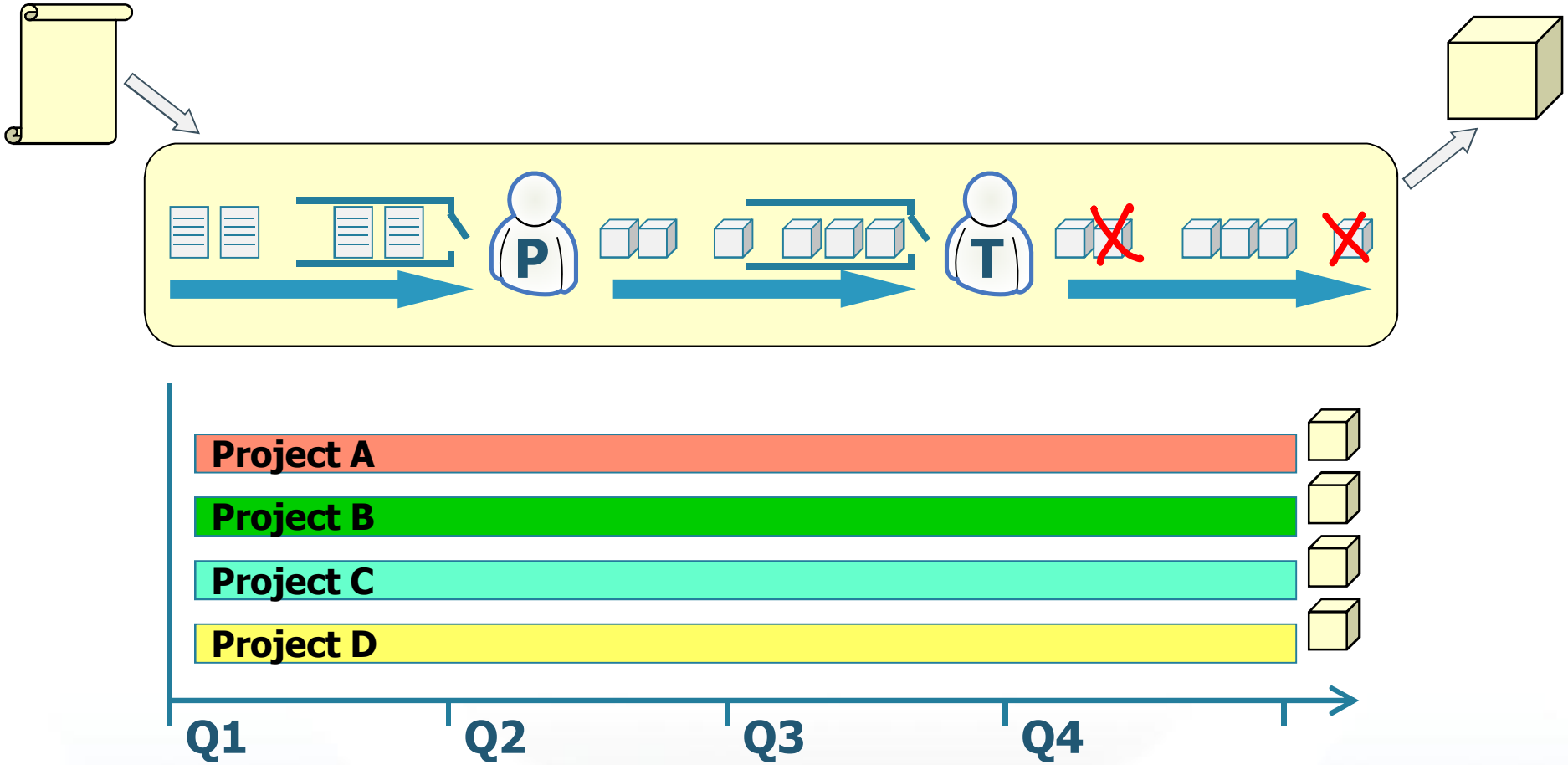
Focusing



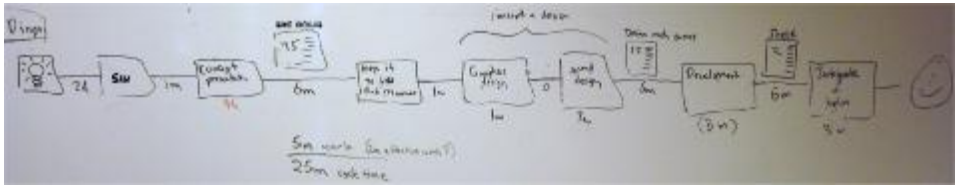
Multitasking

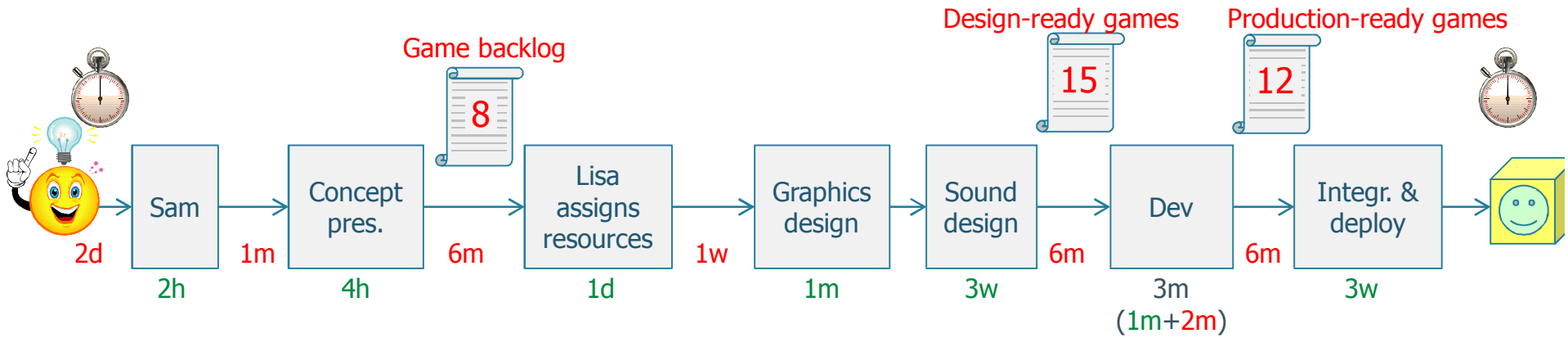


Q: What causes multitasking?
A: Optimizing for max resource utilization



Value stream mapping





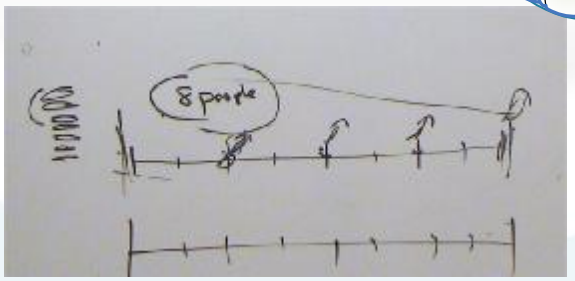
$$\frac{\text{3 m value added time}}{\text{25 m cycle time}} = 12\%$$ Process cycle efficiency

Cross-functional game team

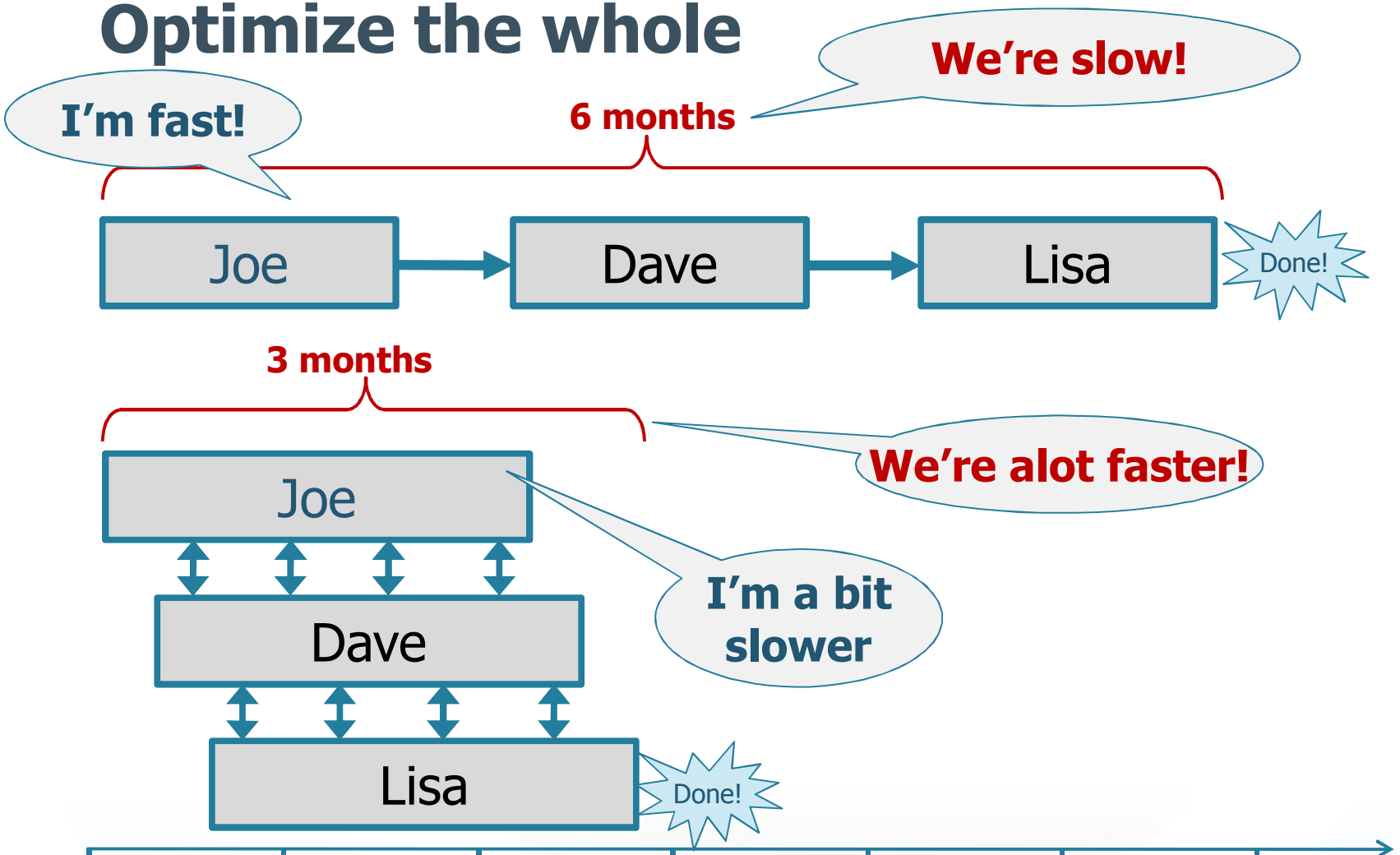
3-4 m cycle time = 6-8x faster

Game team (graphics, sound, dev, integrate)

3-4 months



Optimize the whole



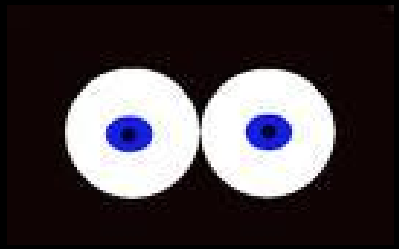
Problem: Invisible baton

Team 1

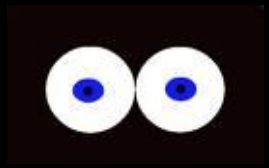
Team 2

Russel Ackoff

Managers who don't know how to
measure what they want
settle for
wanting what they can measure



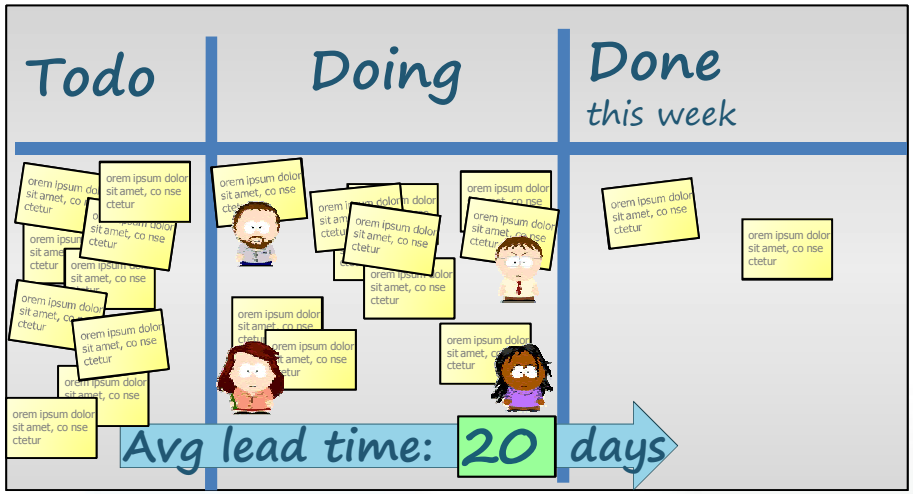
It’s hard to optimize something that you can’t see

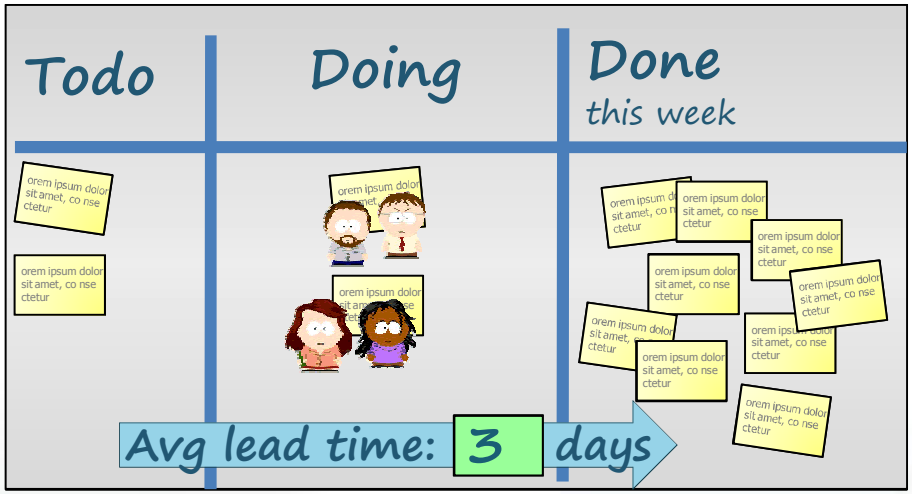


First step: Visualize the baton!

Team 1

Team 2





Kaizen

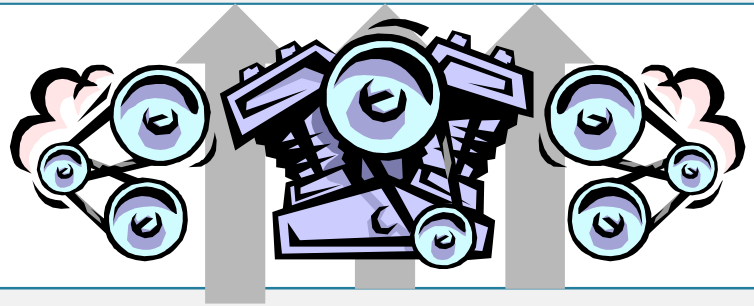
Both are empirical

Capacity
(aka velocity)

Lead time
(aka cycle time)

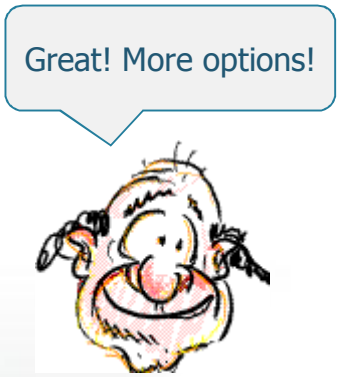
Quality
(defect rate, etc)

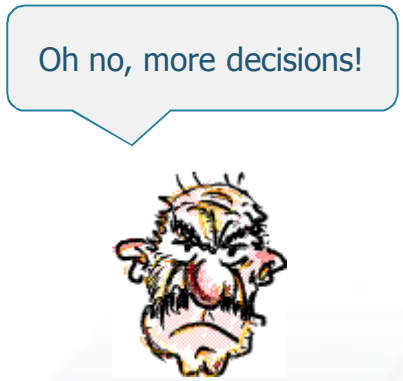
Predictability
(SLA fulfillment, etc)



Many small teams	Few large teams
Low WIP limits	High WIP limits
Few workflow states	Many workflow states
Short iterations	Long iterations
Little planning	Lots of planning
.... etc ...	.... etc ...

Kanban is HIGHLY configurable





Almost always a good idea

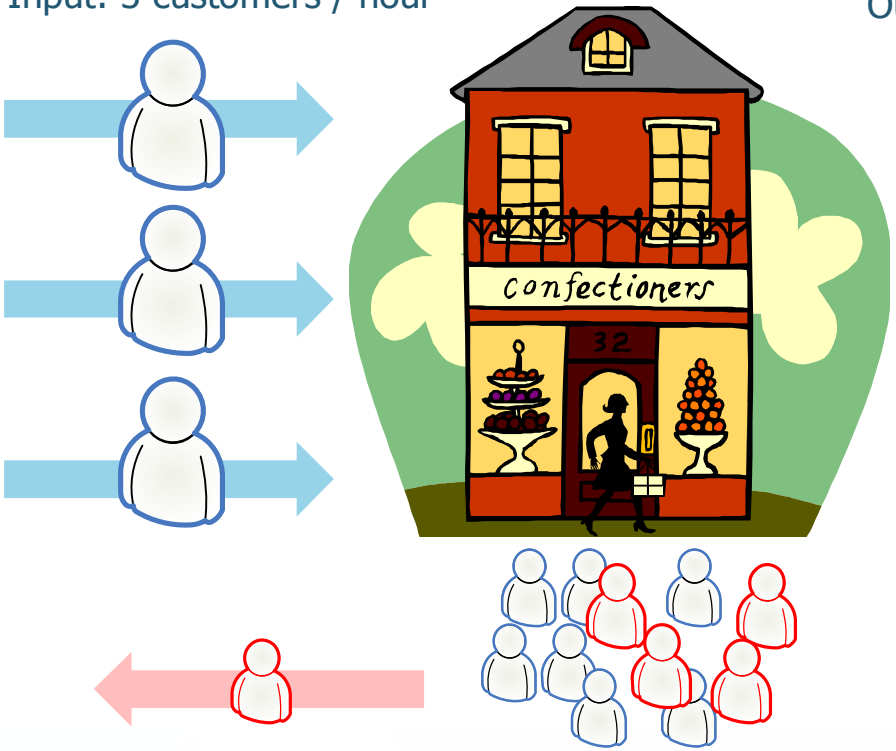
- Visualize the workflow
- Limit WIP (work in process)
- Short feedback loops
- Focus on quality
- Cadences

Exercise

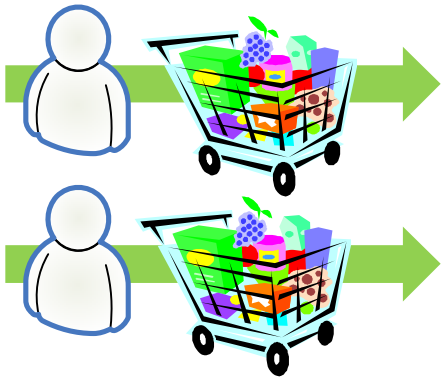
Pull

Limit work to capacity

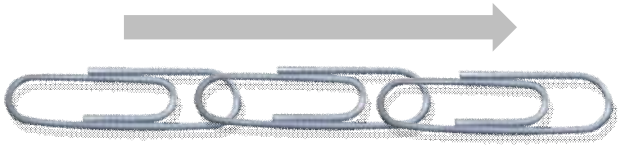
Input: 3 customers / hour



Output capacity: 2 customers / hour

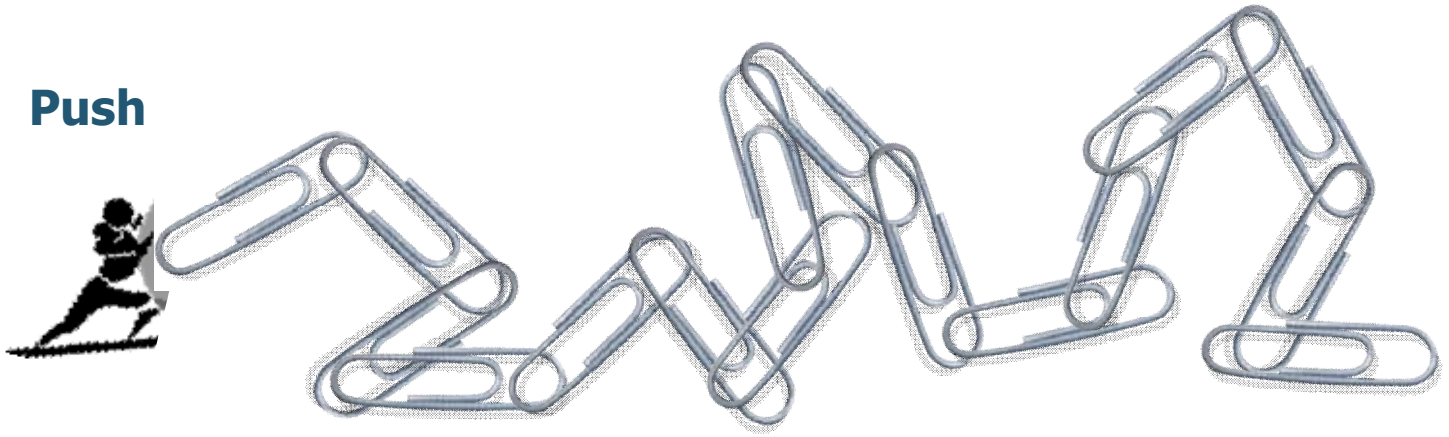


Pull vs Push

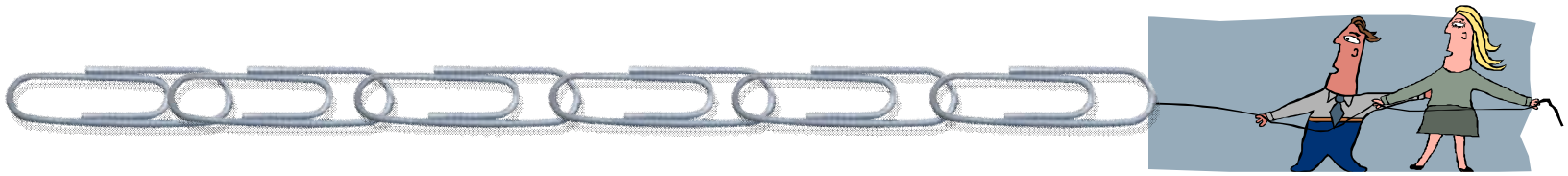


Assigning tasks = push
Taking tasks = pull

Push

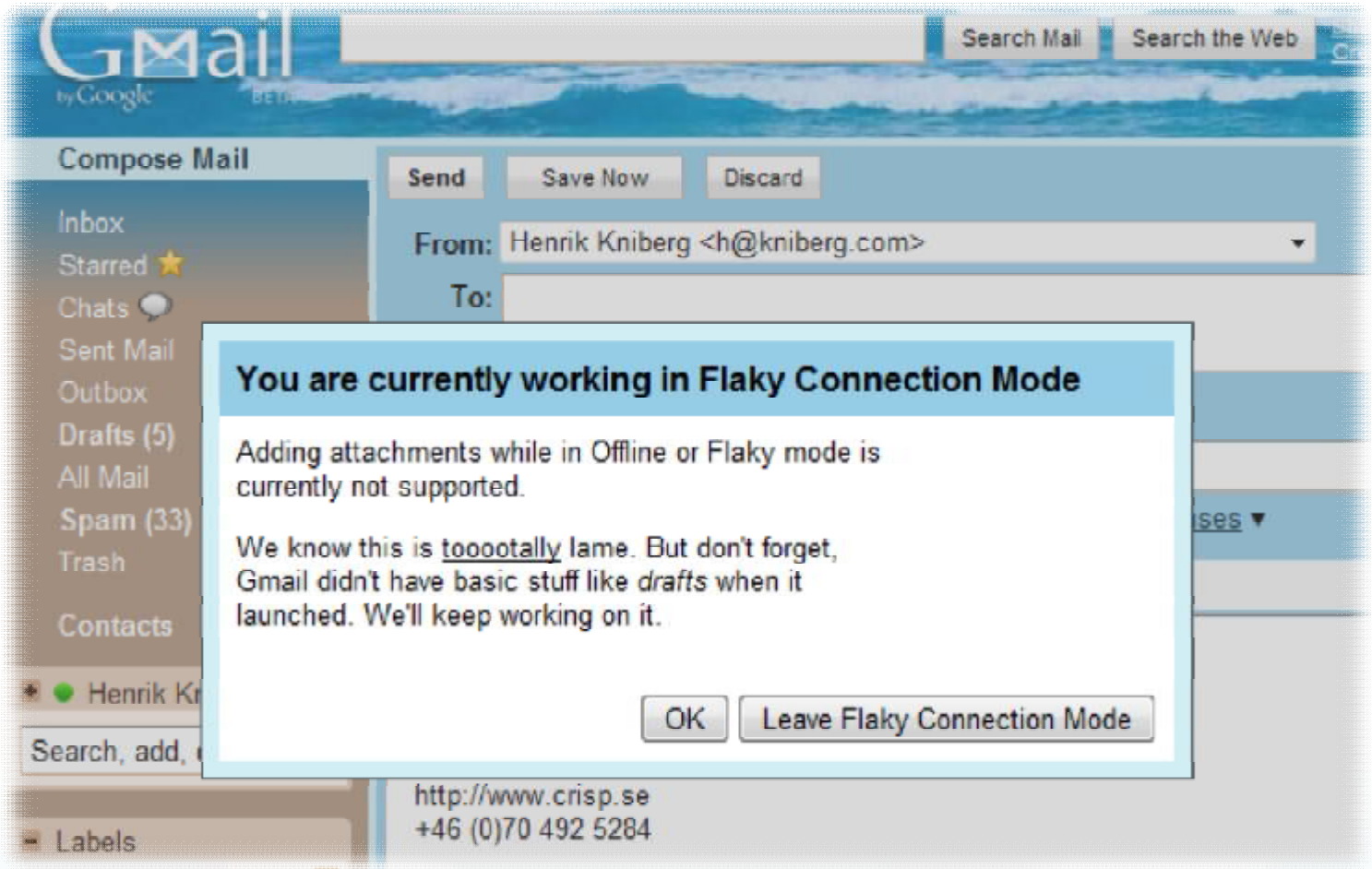


Pull



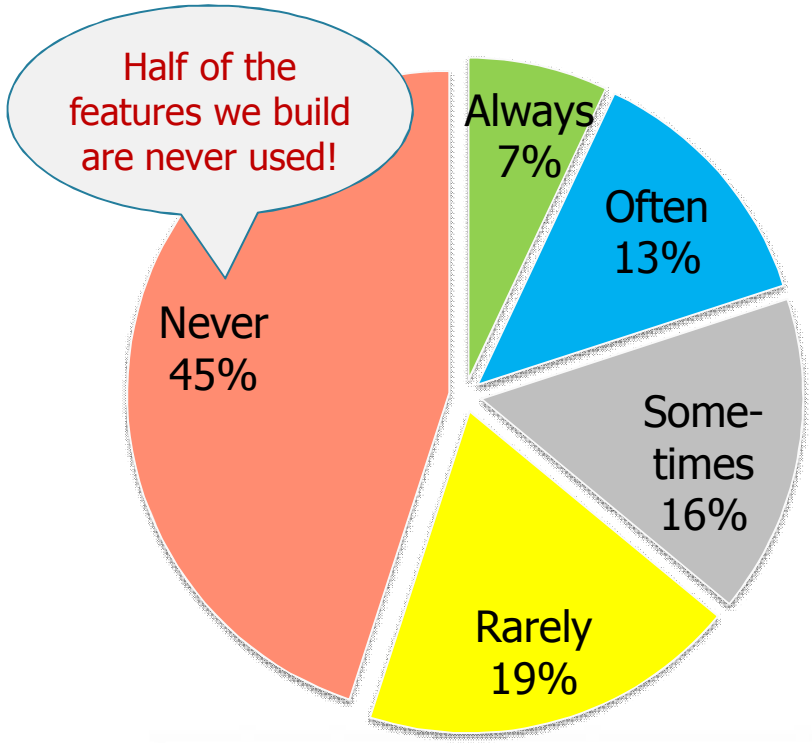
Example: Supermarket pull

Example: Feature pull



Why feature pull is important

Features and functions used in a typical system:



Source: Standish Group Study Reported at XP2002 by Jim Johnson, Chairman

Kanban

Kanban

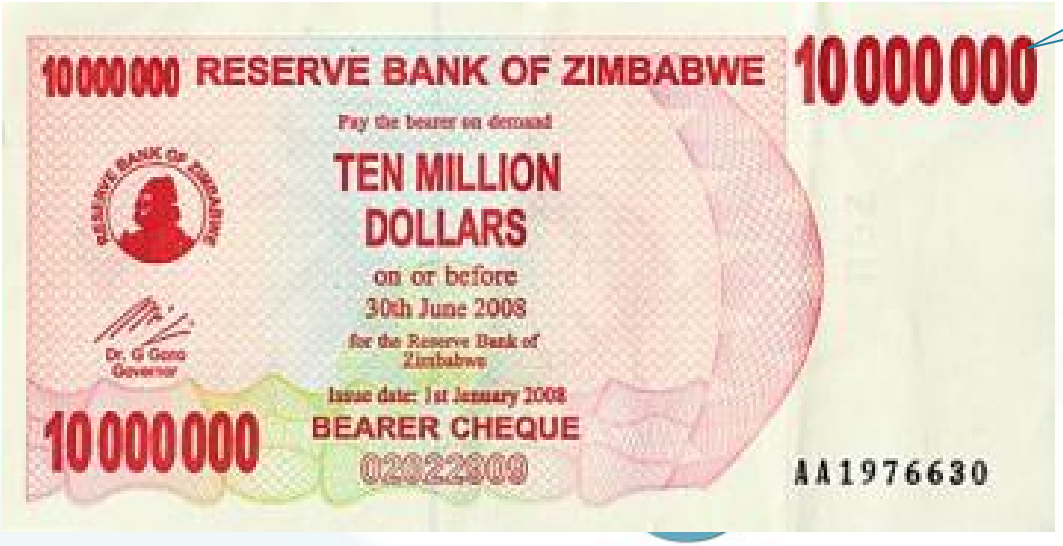
看板

”Visual Card”

- **Signaling system**
- **Visual**
- **Limited in supply**

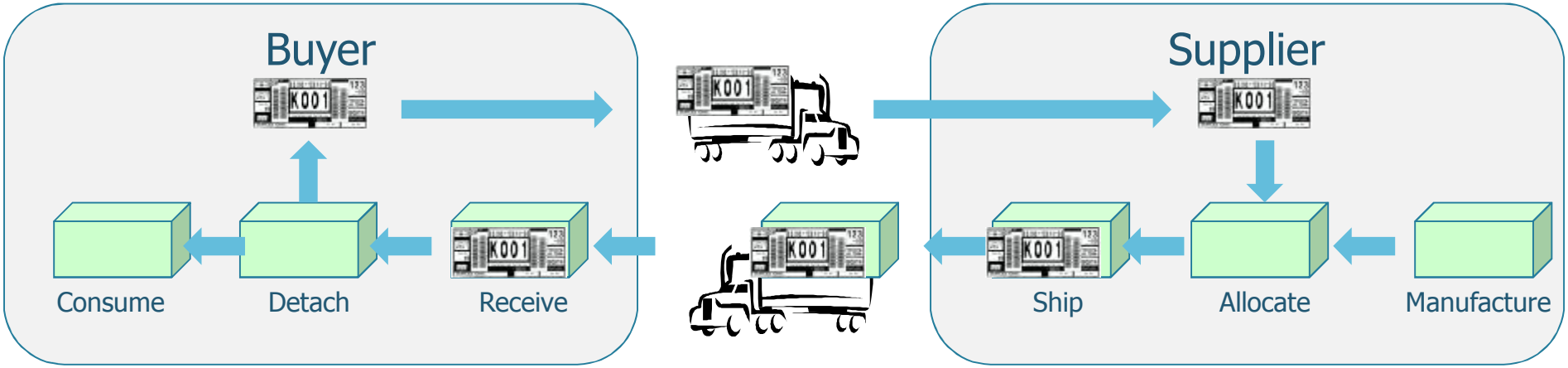
Kanban @ Imperial Palace Gardens

Kanban in your wallet



Darn. Forgot to limit.

Kanban @ Toyota



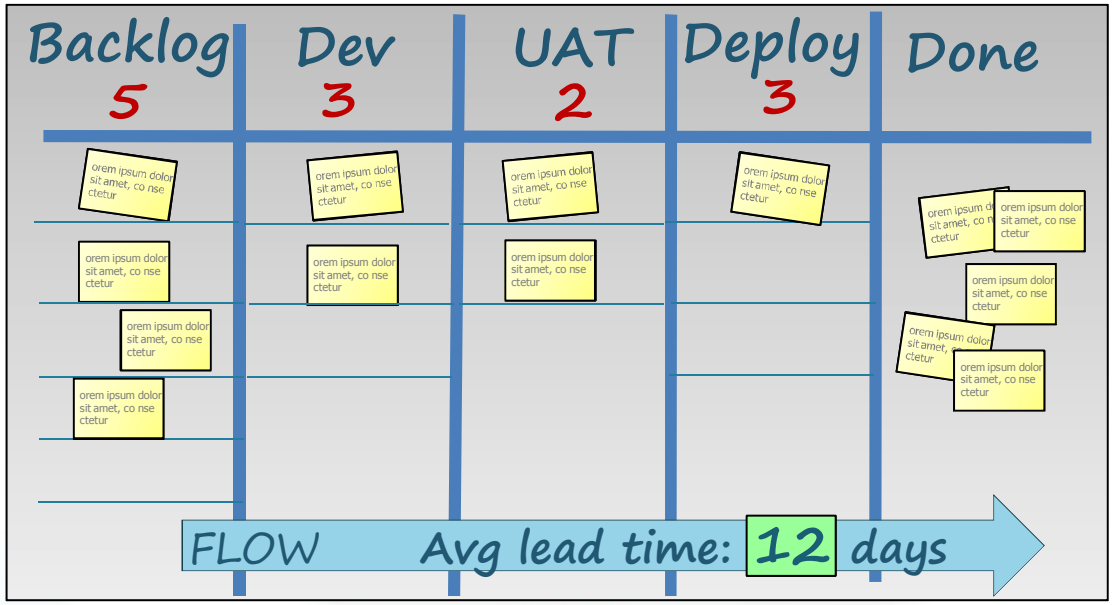
The tool used to operate the [Toyota Production] system is kanban.

Taiichi Ohno
Father of the
Toyota Production System

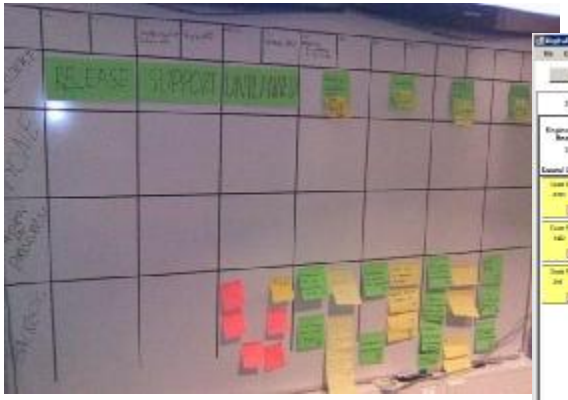
Kanban in SW development

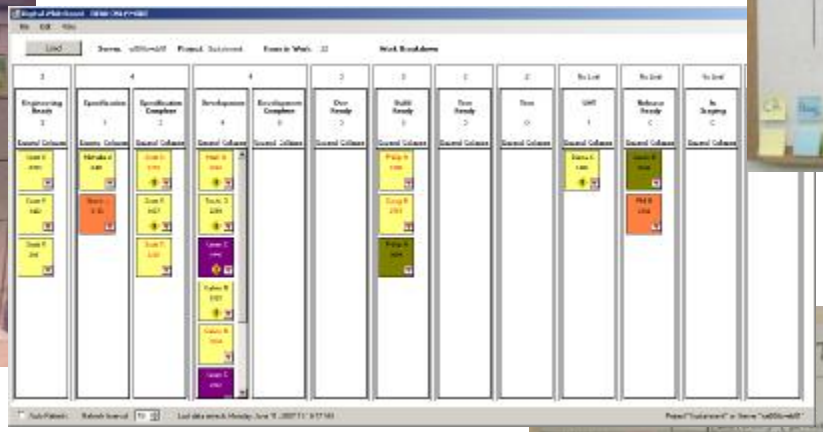
Pioneered by David Anderson in 2004

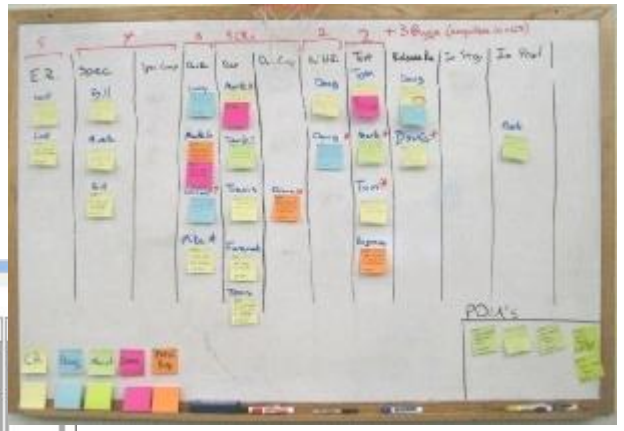
- **Visualize the workflow**
- **Limit WIP (work in progress)**
- **Measure & optimize flow**

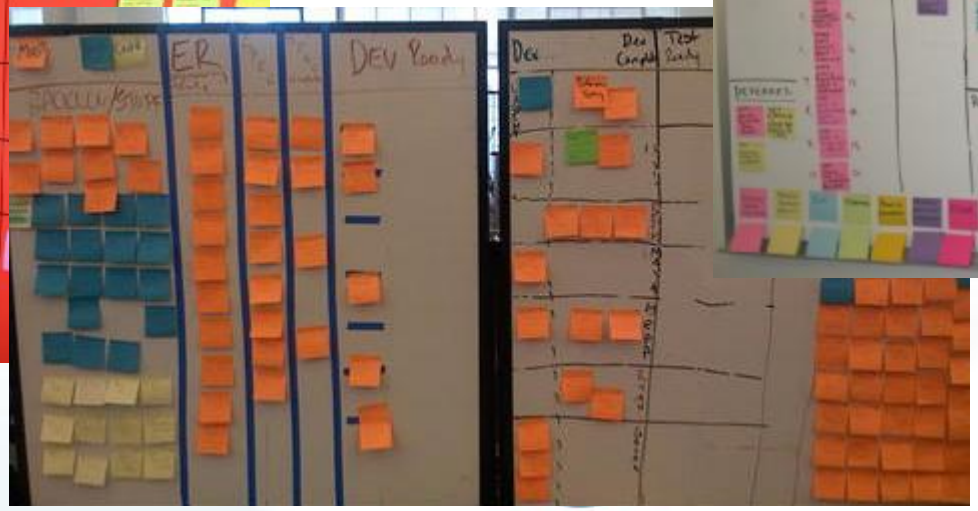


Evolve your own board!







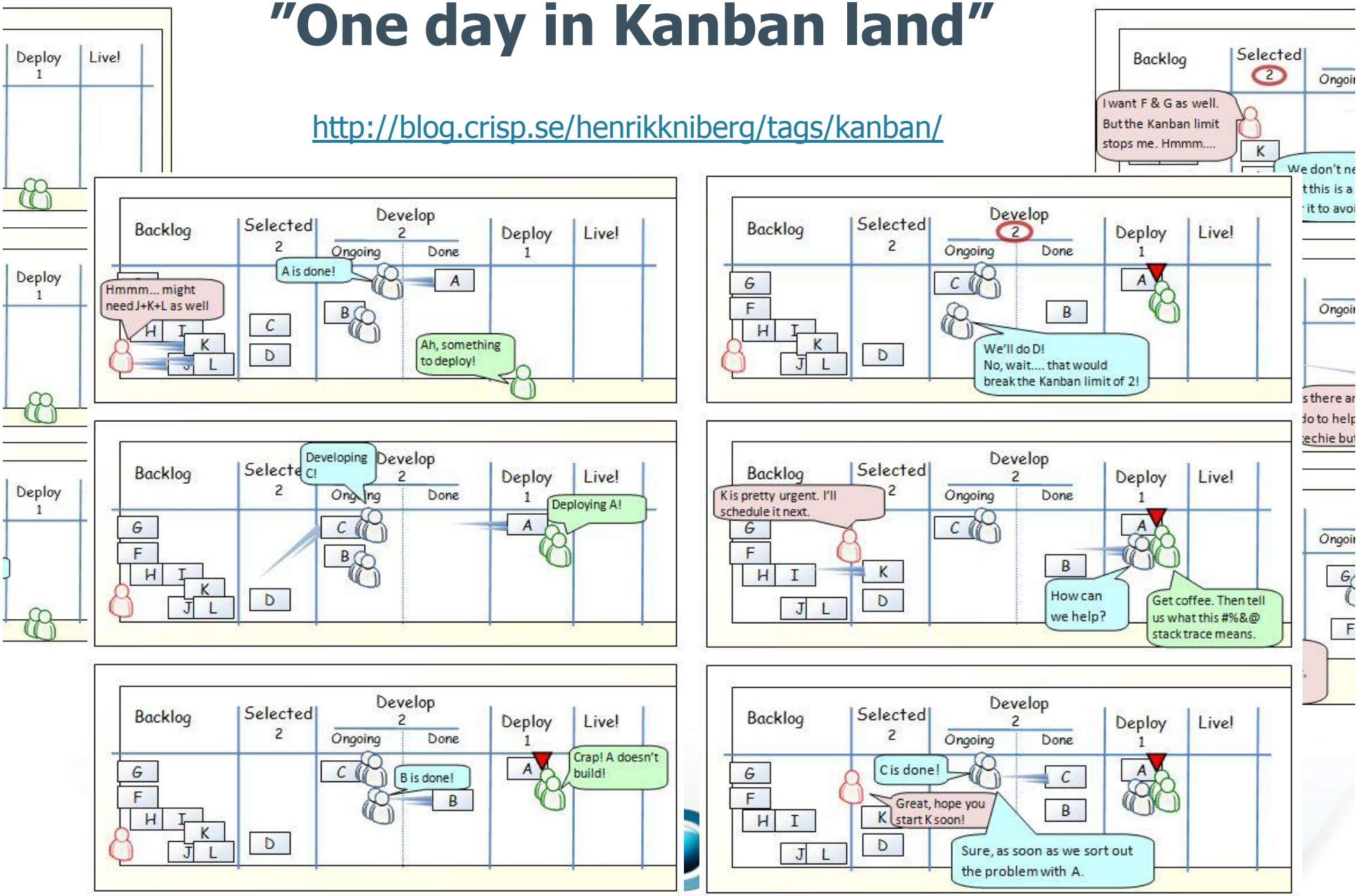


Some of these photos courtesy of David Anderson, Mattias Skarin, and various other people

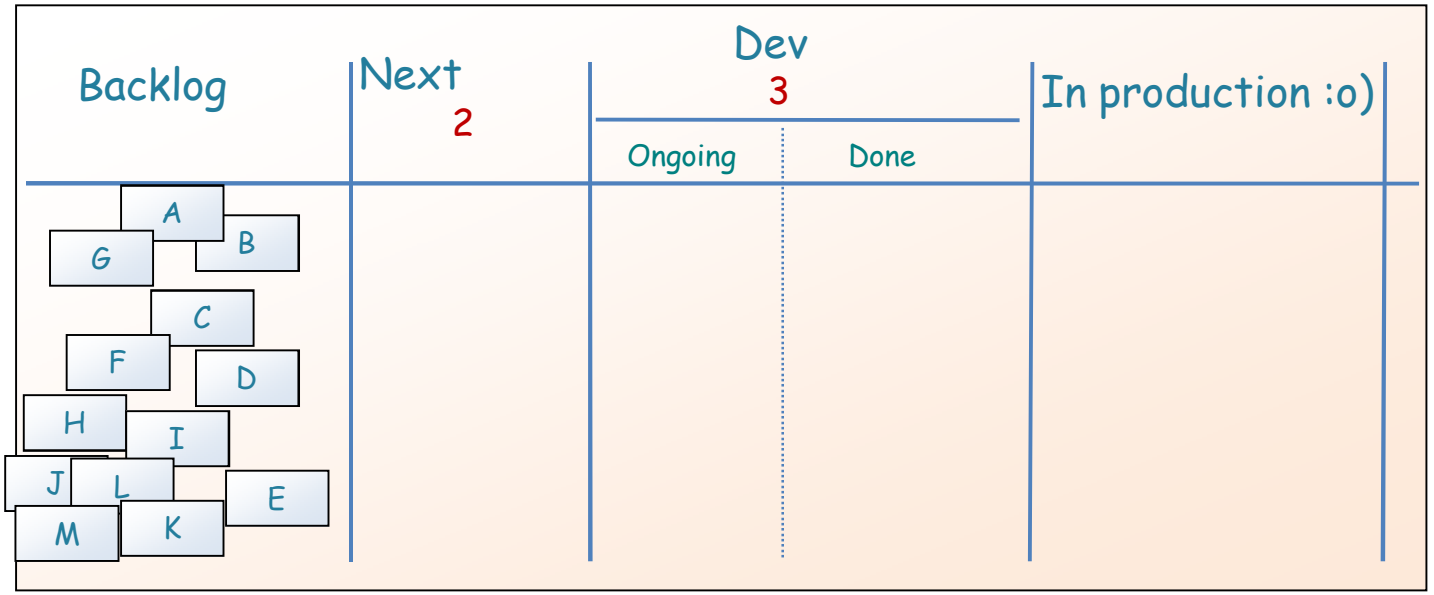
One day in Kanban land

”One day in Kanban land”

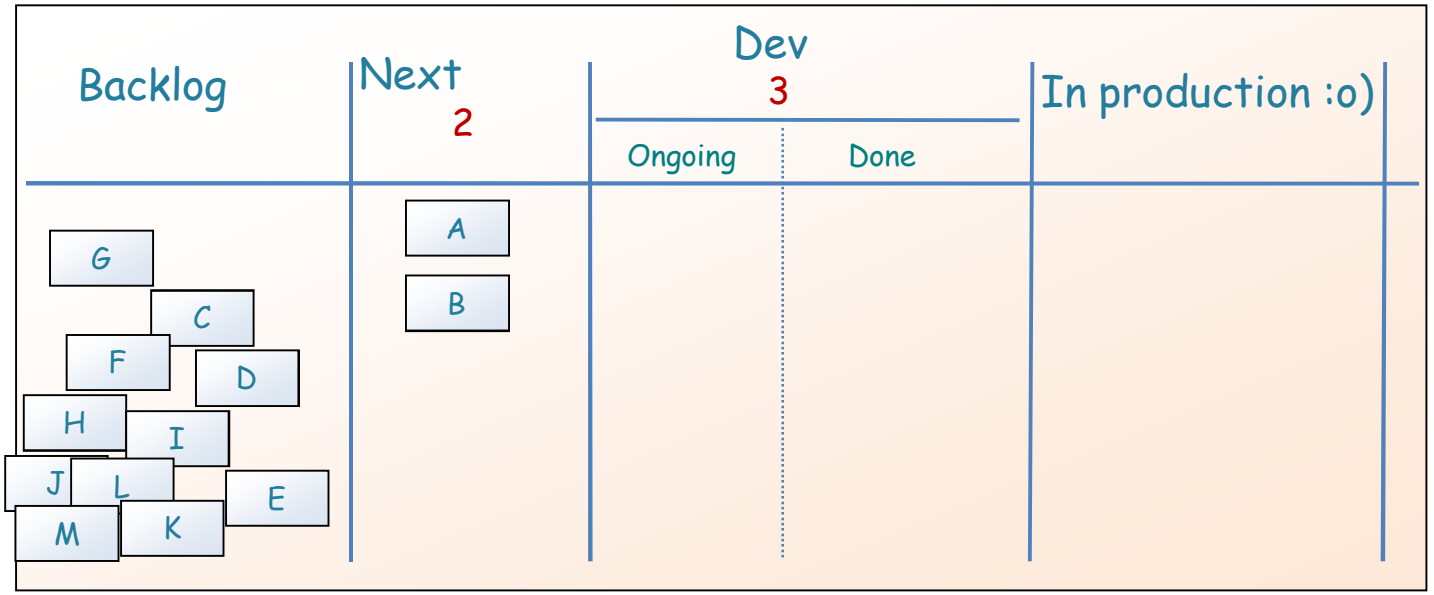
http://blog.crisp.se/henrikkniberg/tags/kanban/



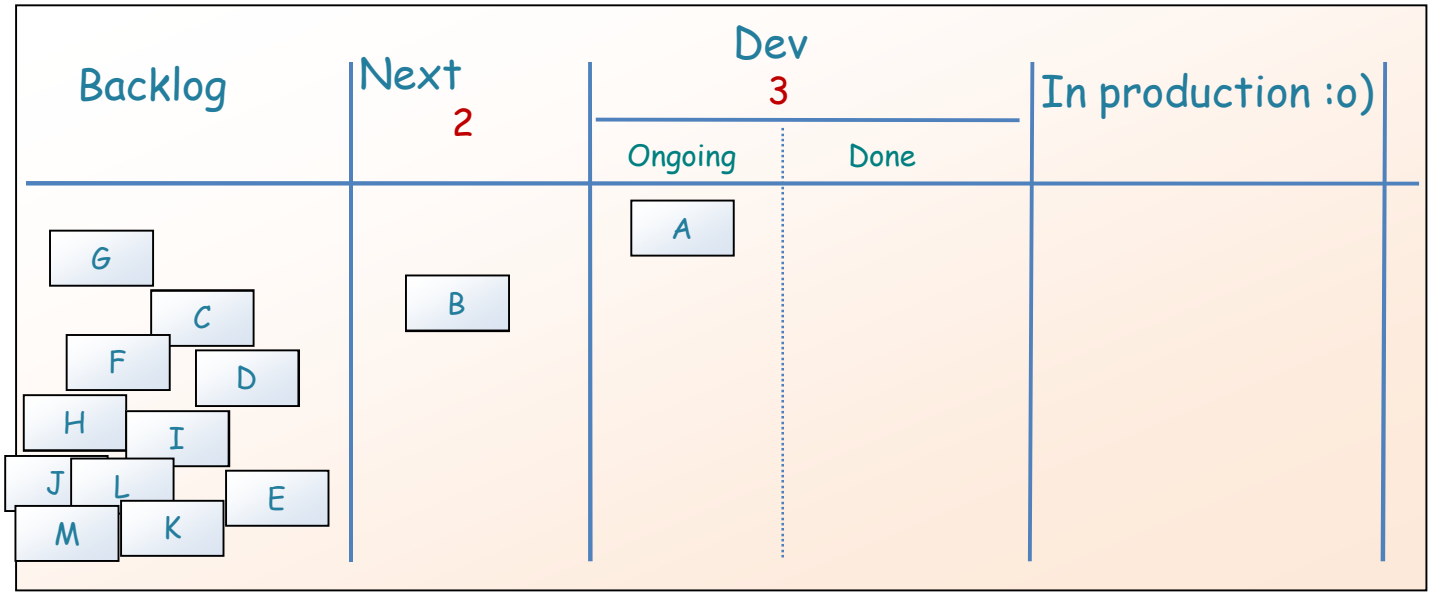
Scenario 1 – one piece flow



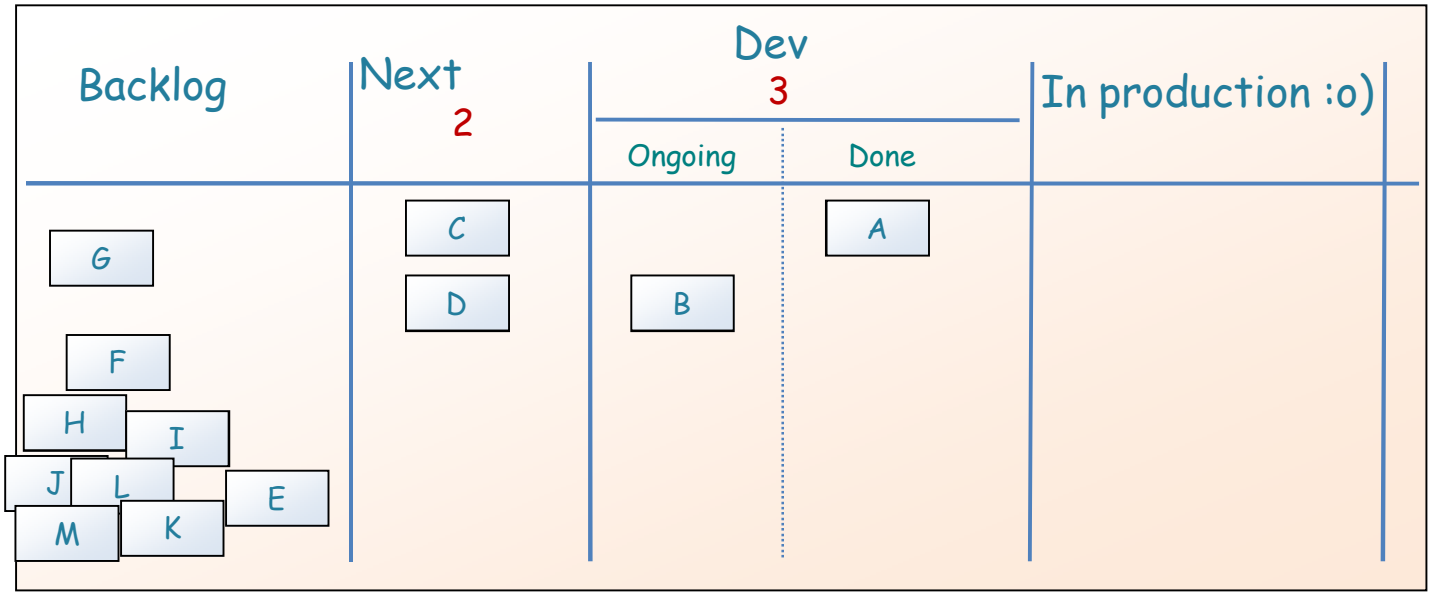
Scenario 1 – one piece flow



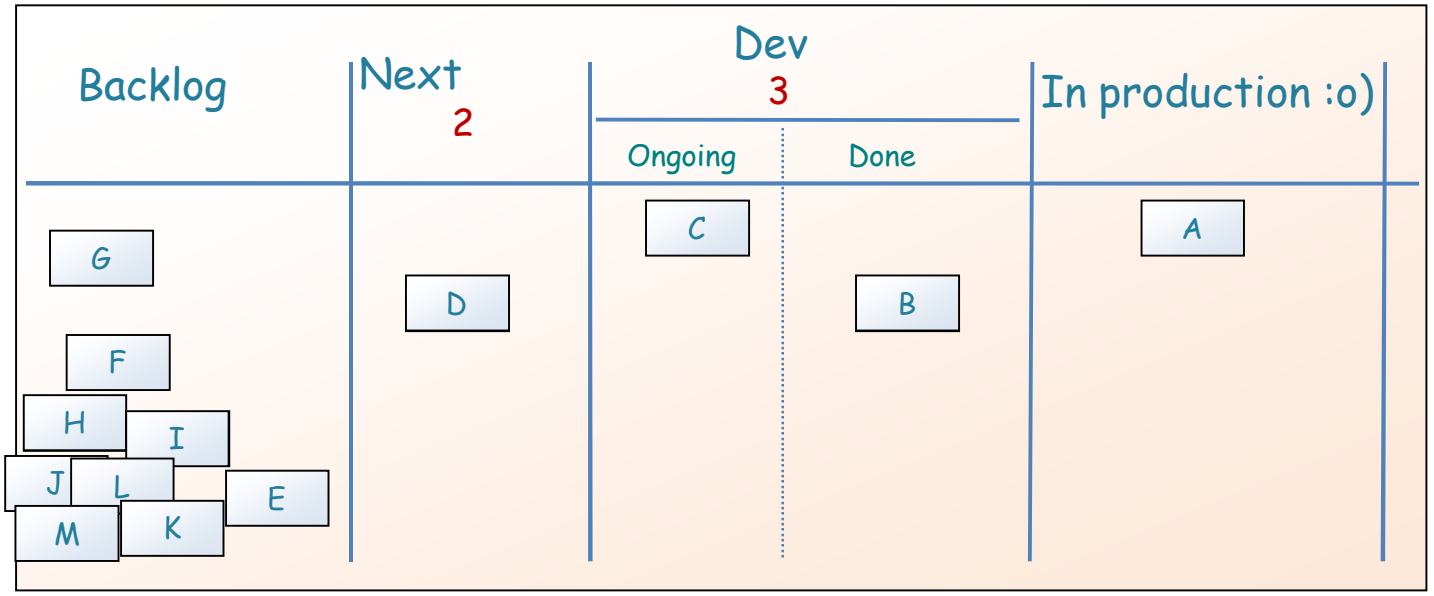
Scenario 1 – one piece flow



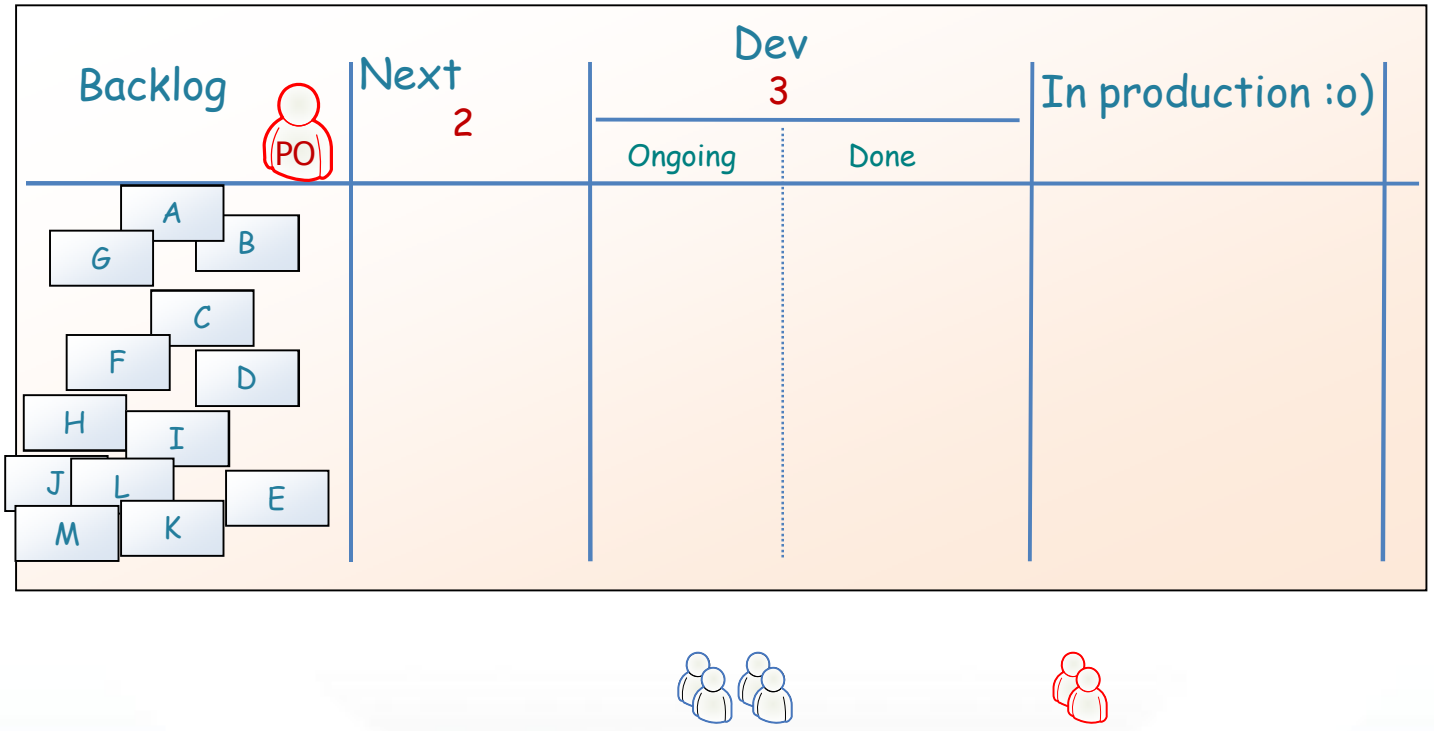
Scenario 1 – one piece flow



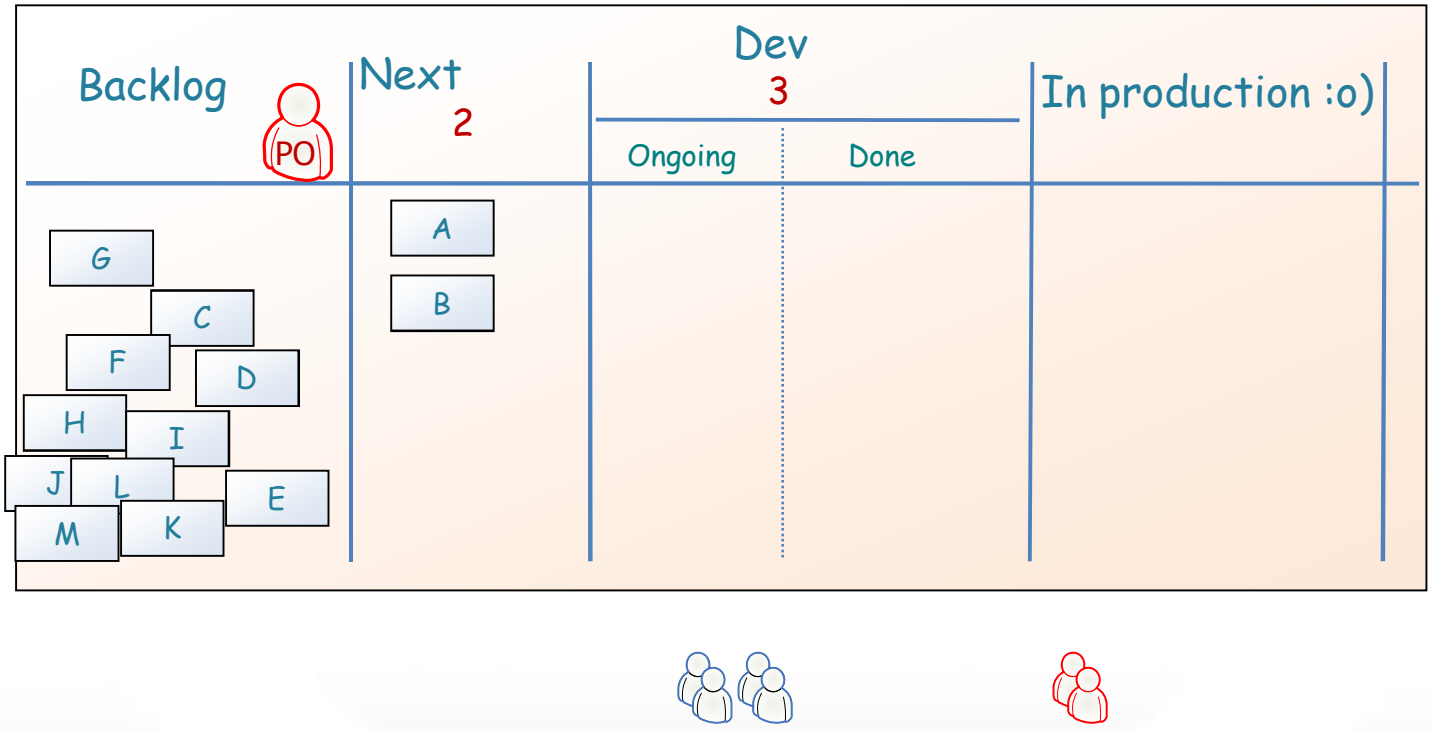
Scenario 1 – one piece flow.



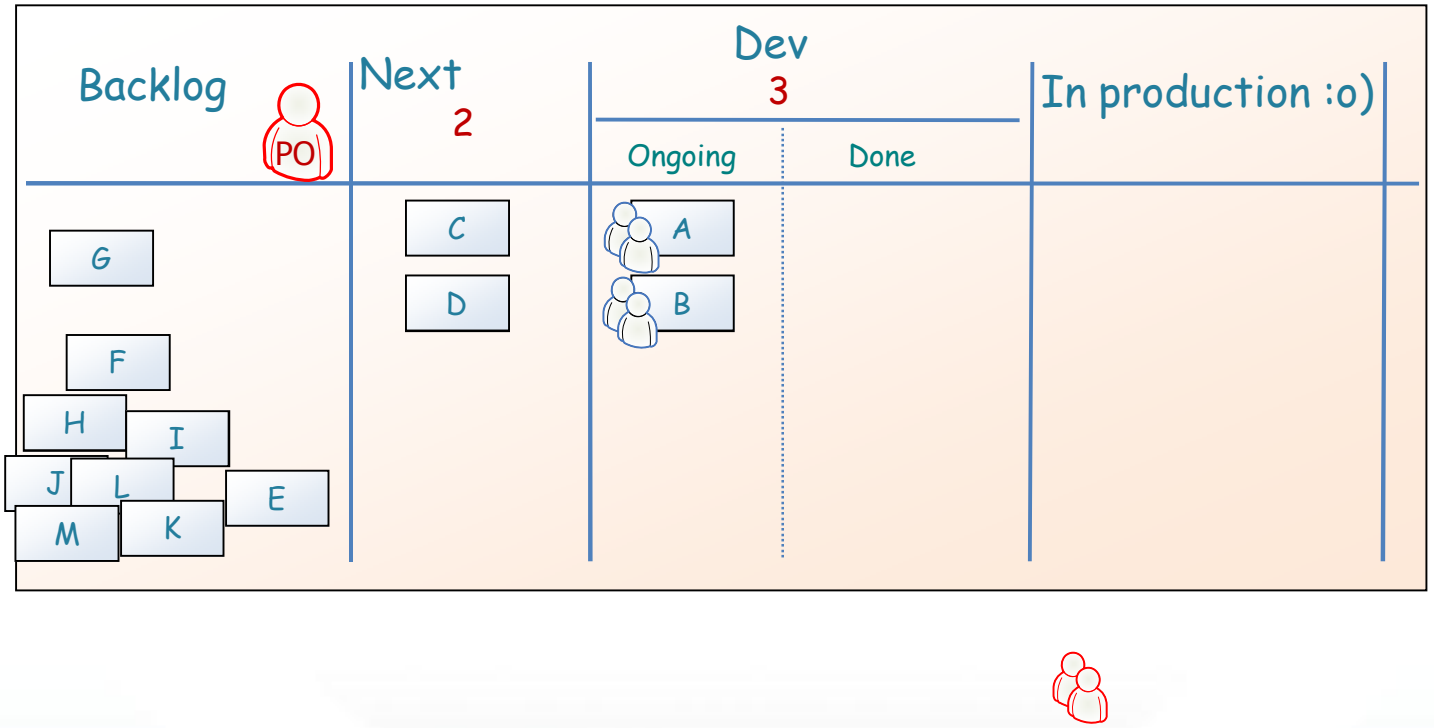
Scenario 2 – Deployment problem



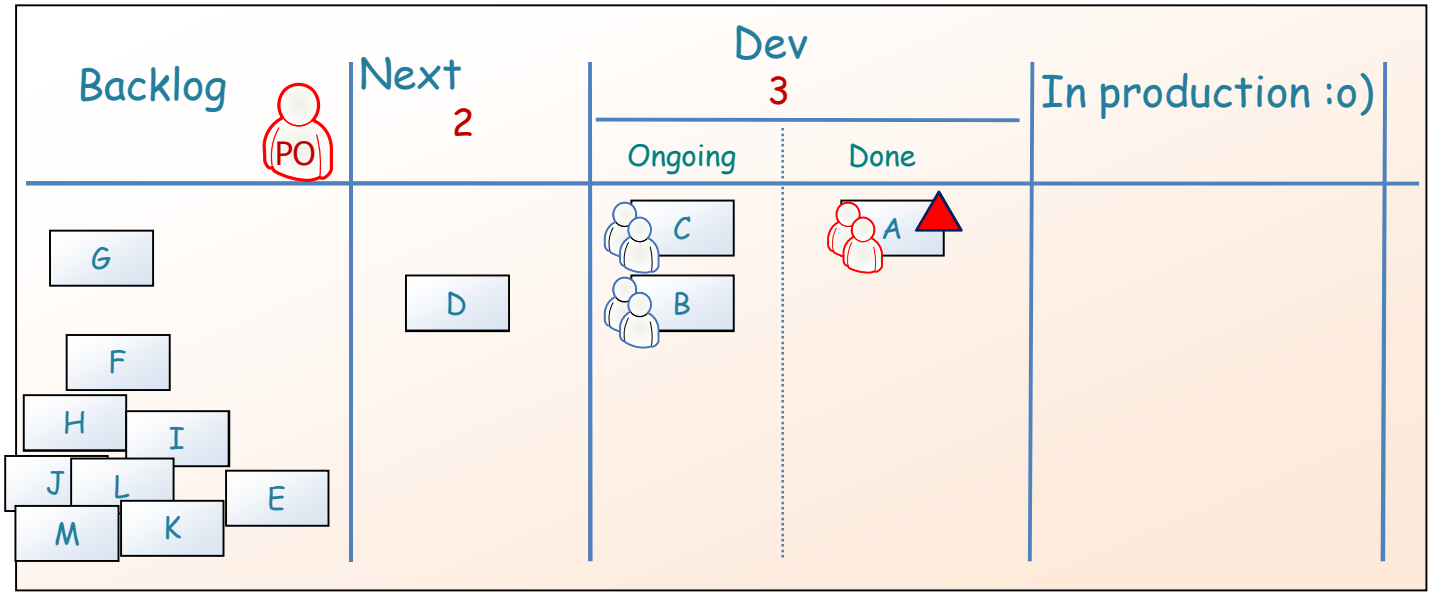
Scenario 2 – Deployment problem



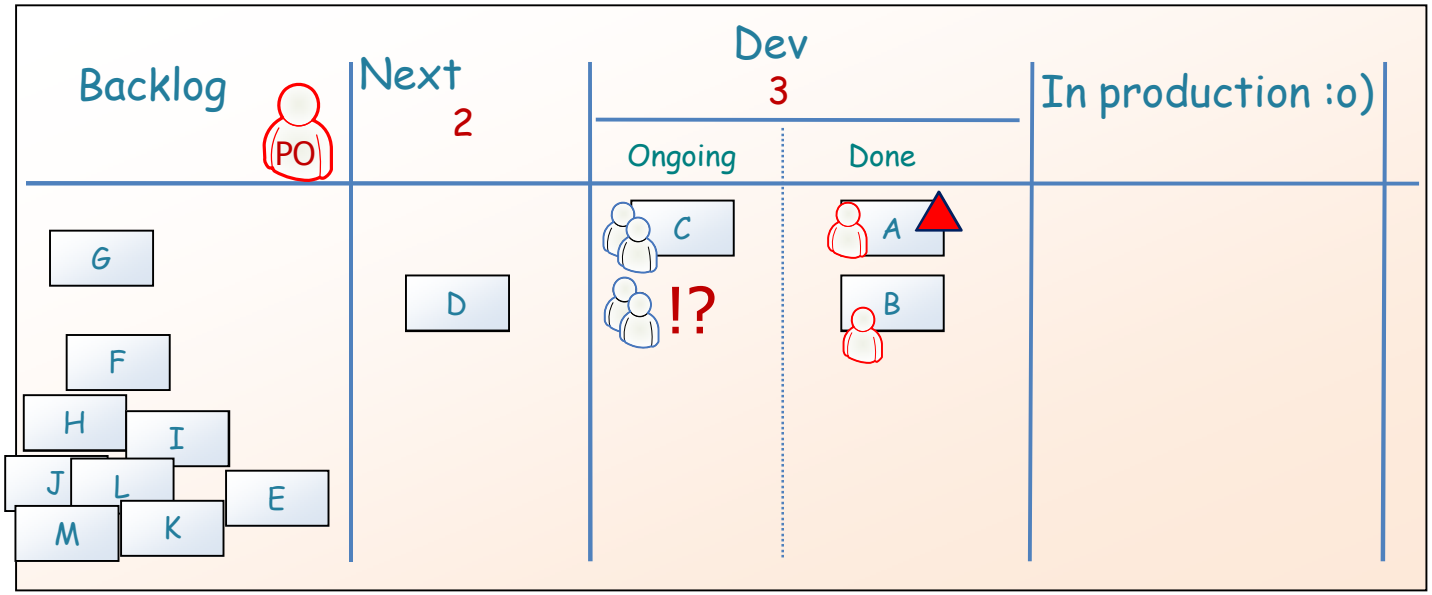
Scenario 2 – Deployment problem



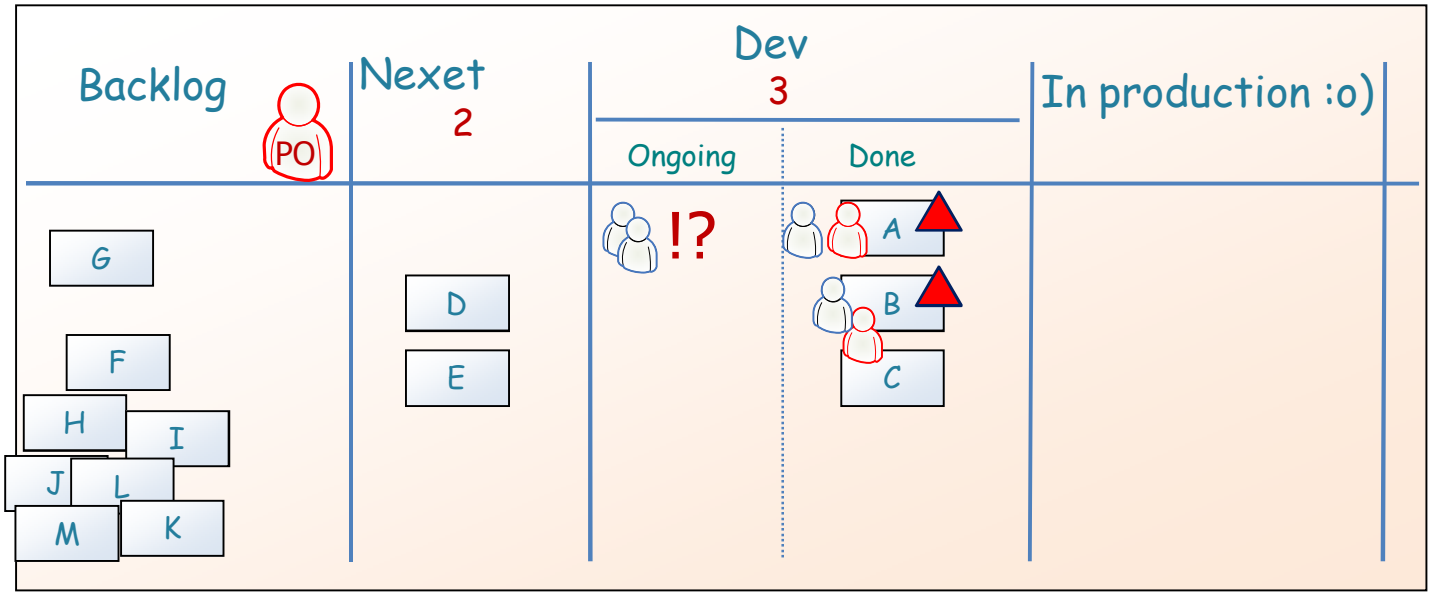
Scenario 2 – Deployment problem



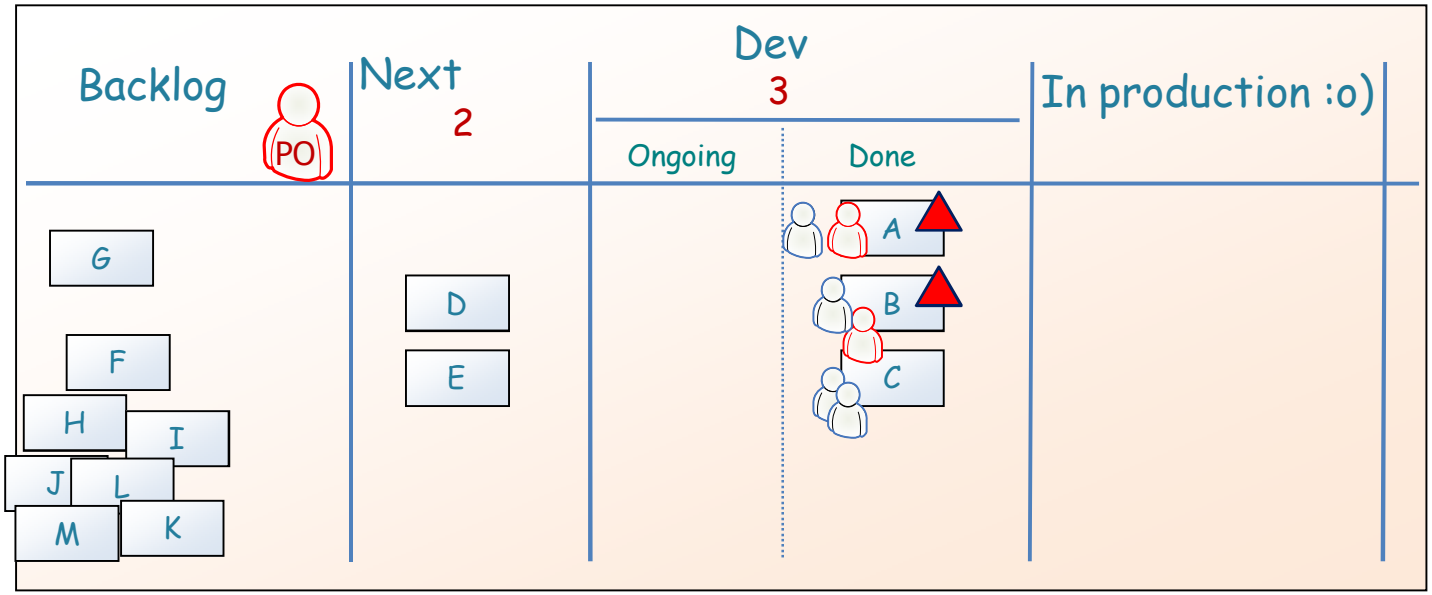
Scenario 2 – Deployment problem



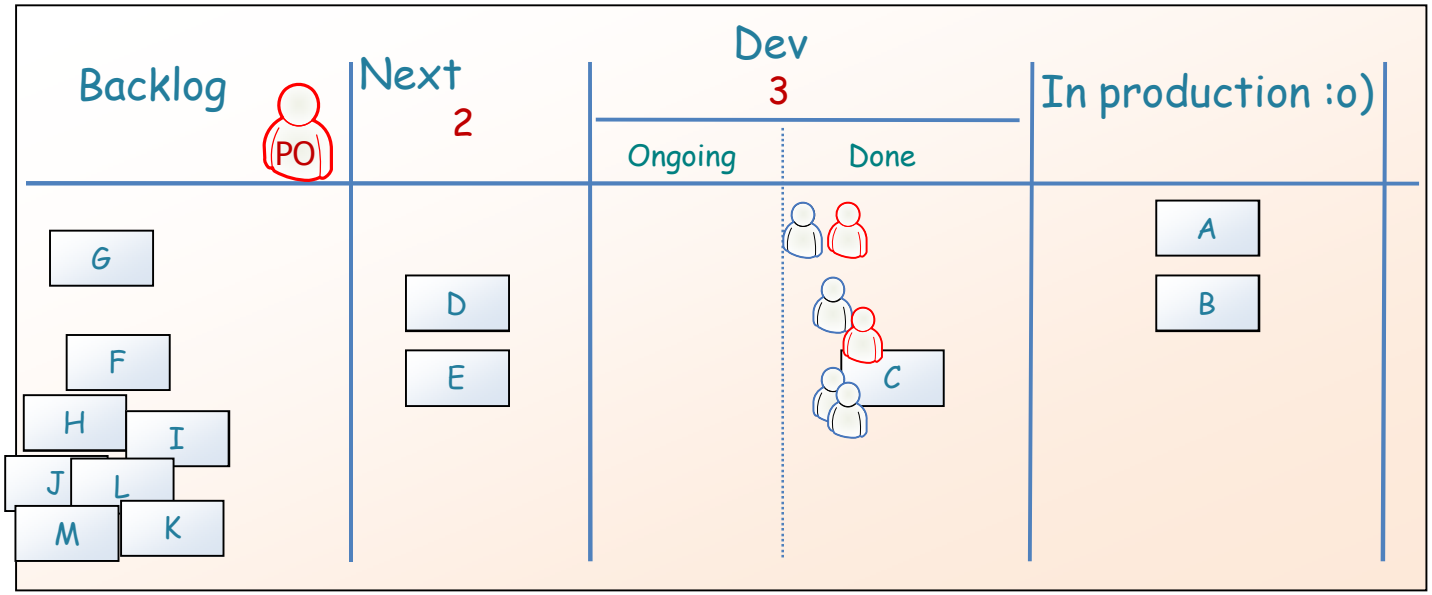
Scenario 2 – Deployment problem



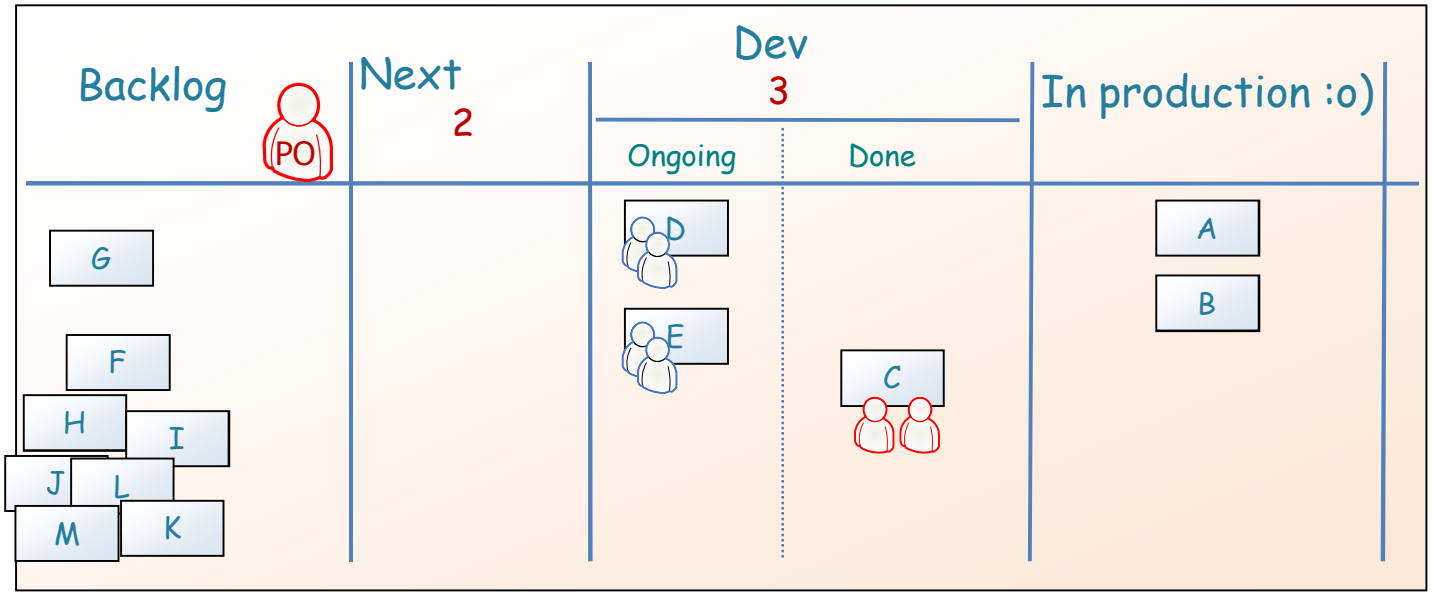
Scenario 2 – Deployment problem



Scenario 2 – Deployment problem

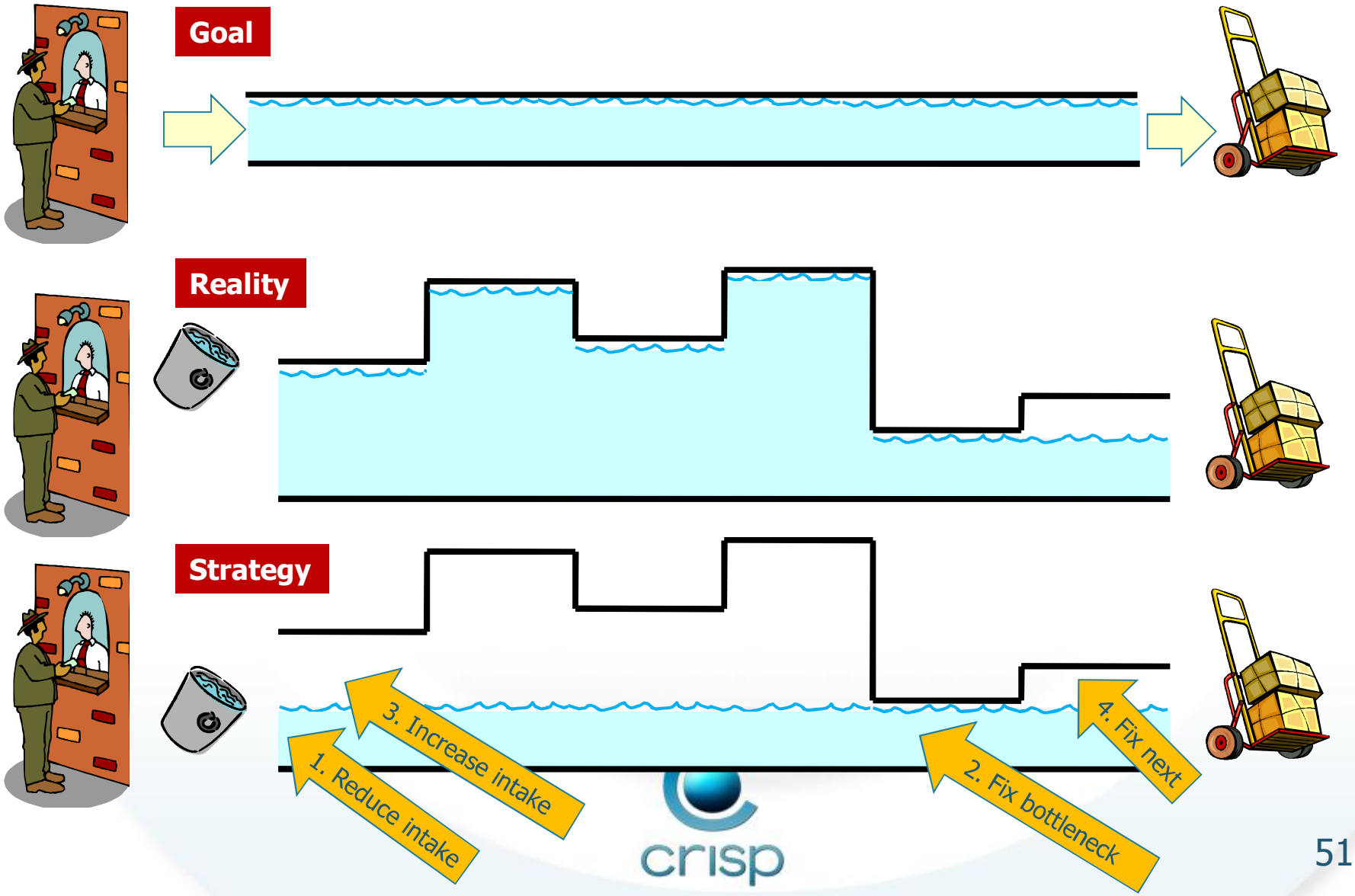


Scenario 2 – Deployment problem



Bottlenecks

Theory of constraints – Smooth Flow



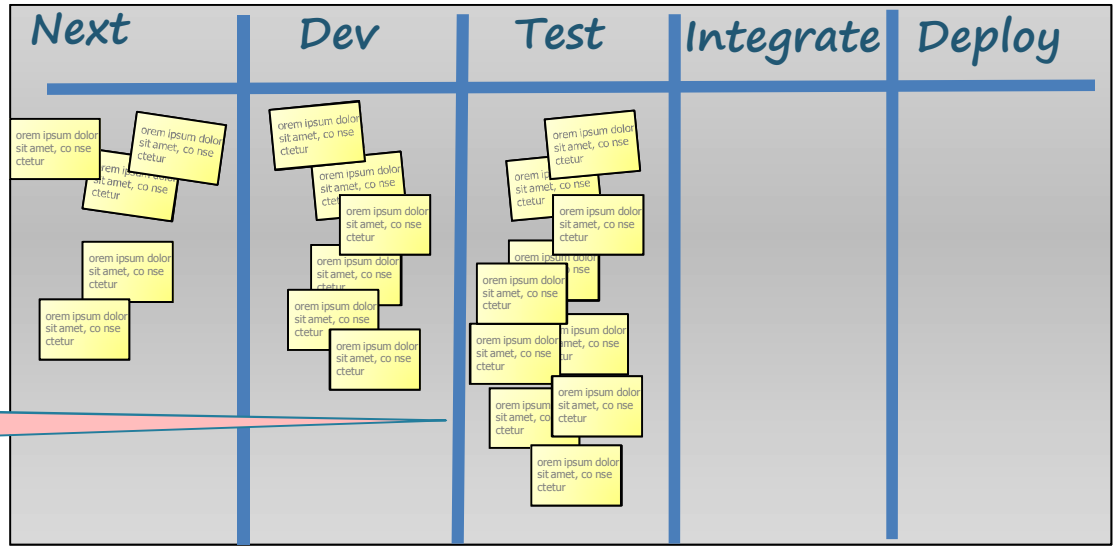
Theory of constraints

- **All systems have a bottleneck**
- **When you fix it, it jumps to somewhere elese.**
- **The whole system performs only as well as its bottleneck**
- **Ensure you have slack before & after the bottleneck**
  - Happens automatically with pull scheduling
- **Use the slack to fix the bottleneck**

How to find your bottleneck

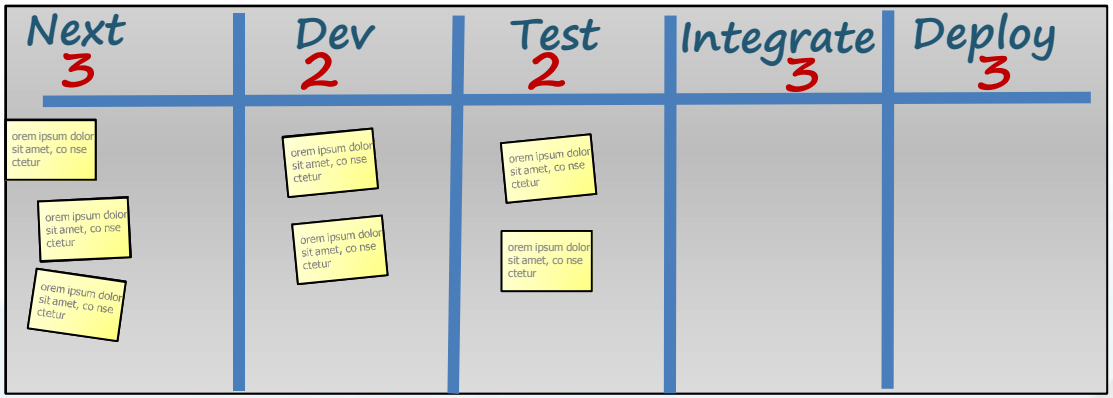
- Busy at bottleneck
- Slack downstream
- Queue upstream

Without WIP limits

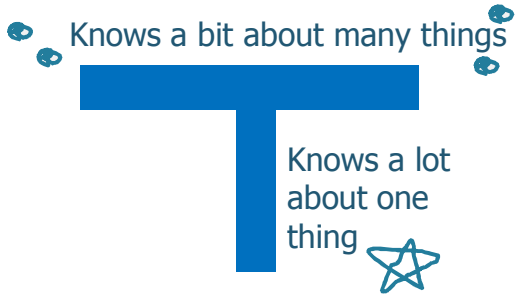


Cheap queue

With WIP limits



Goal: "T-shaped" people



Team Skills Matrix

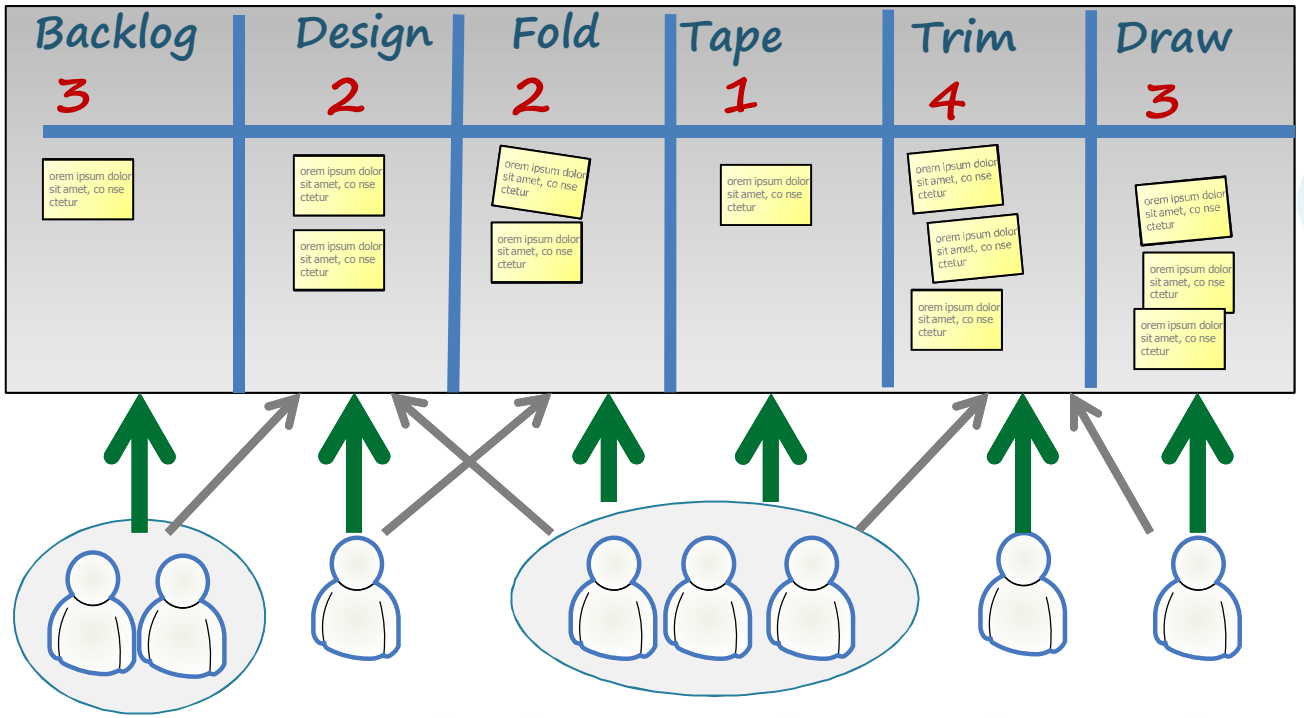
I *can* test, but I'm not so good at it.

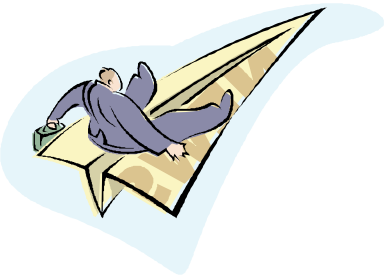
I'm good at Java!

I don't know CM at all. But I'm willing to learn!

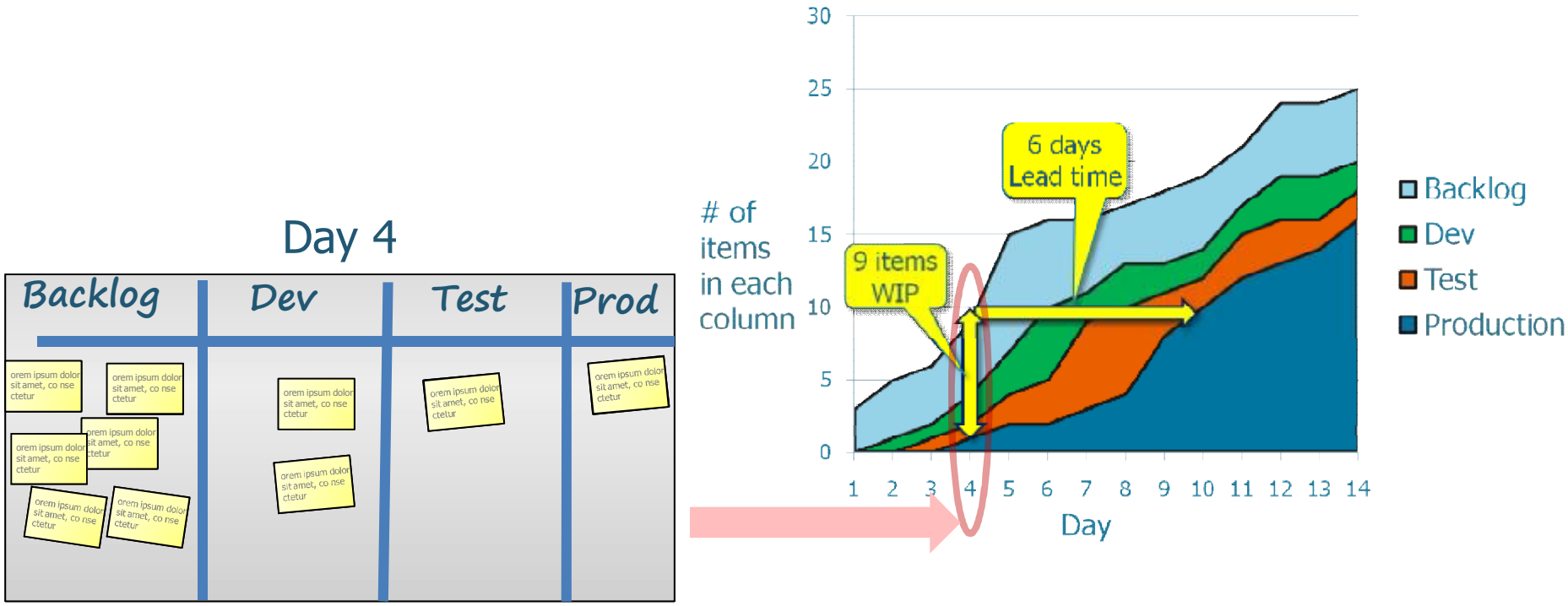
	Test	DB	Web	Java	Domain	CM
Lisa	●	●	●	★		●
Joe	●	★		●	●	
Fred	●			★	●	
Jenny	●		★	●		
David	★		●		★	●
Erik			★	★	●	★

Kanban allows both specialists & generalists





Cumulative Flow Diagram (CFD)



Cadence

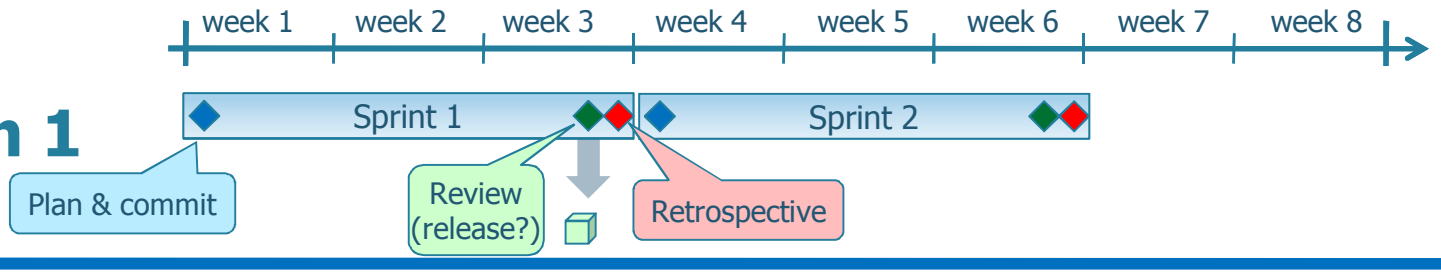
Daily standup meeting

Source: Visual management blog
http://www.xqa.com.ar/visualmanagement/2009/04/daily-scrum-against-the-board/

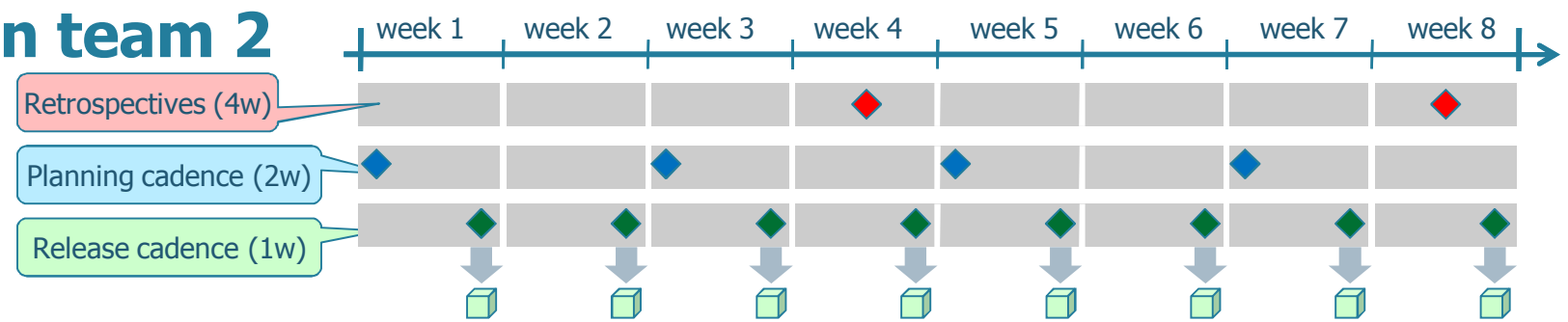
Cadence

Scrum team

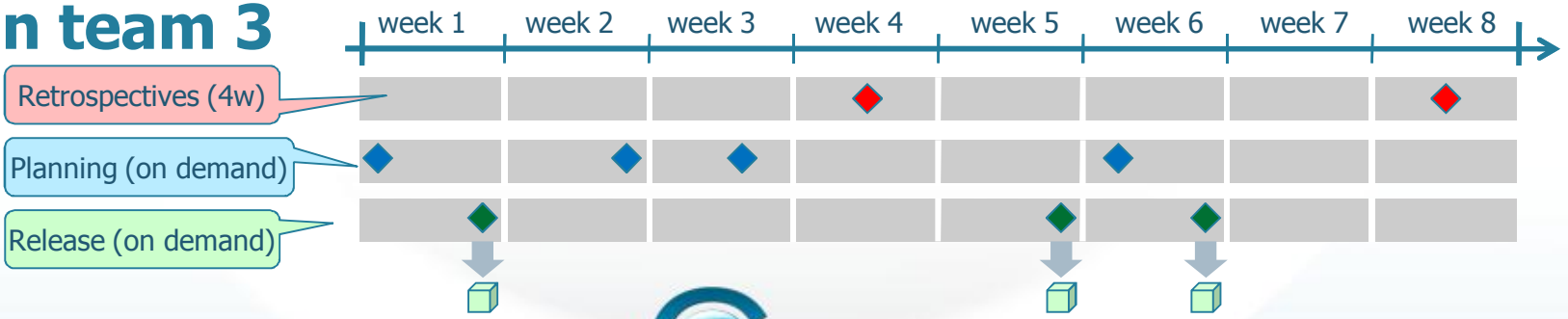
Kanban team 1



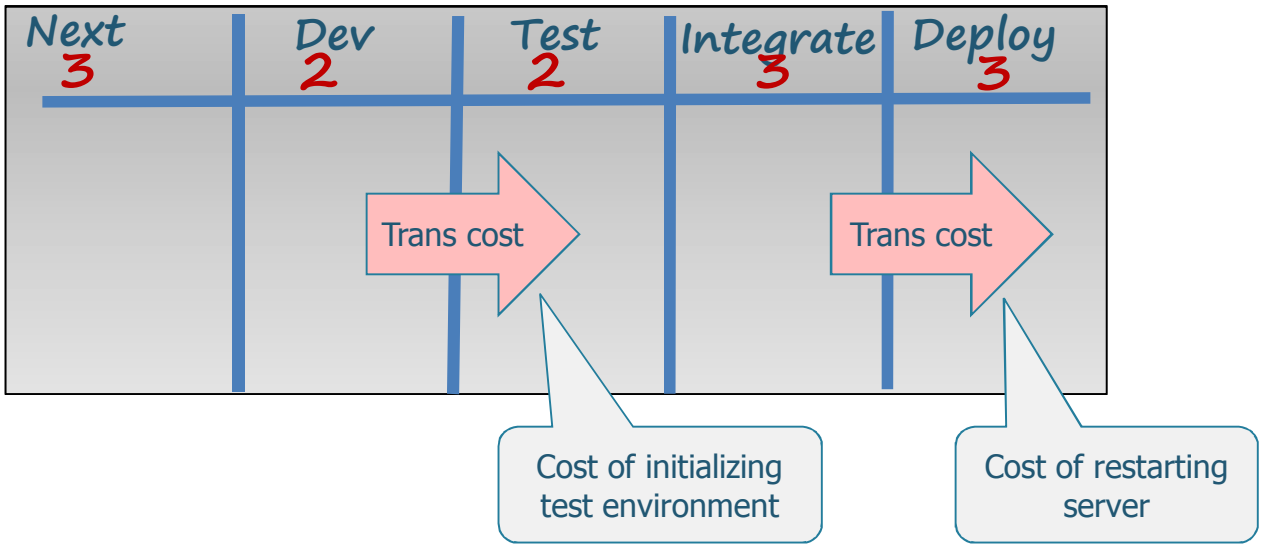
Kanban team 2



Kanban team 3



Reduce transaction costs



Economies of Scale

"We need to do big batches because transaction cost is high"

Economies of Flow (Lean)

"We need to minimize transaction cost so we can do smaller batches"

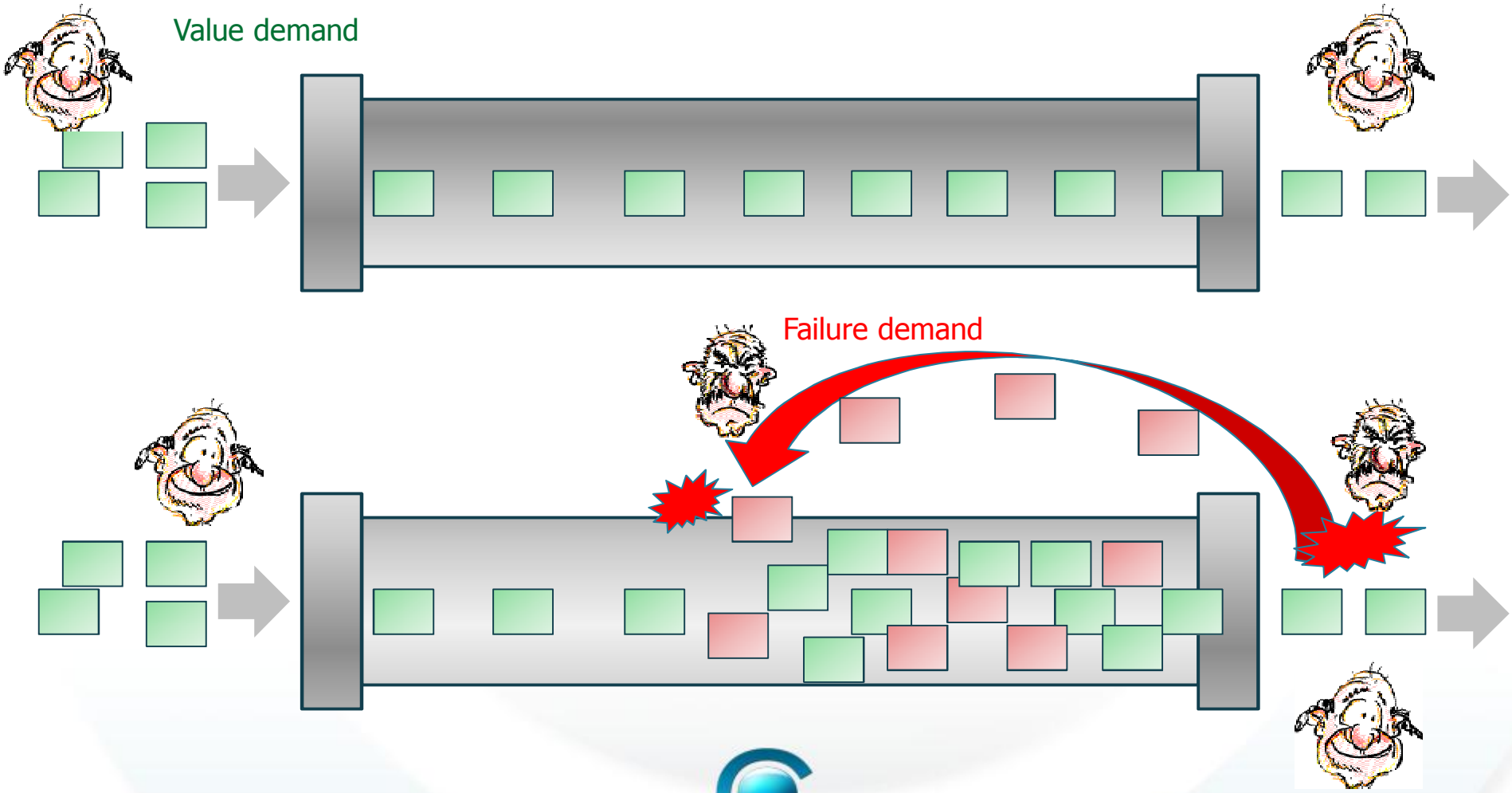
Automate init of test environment

Implement hot-reload on server

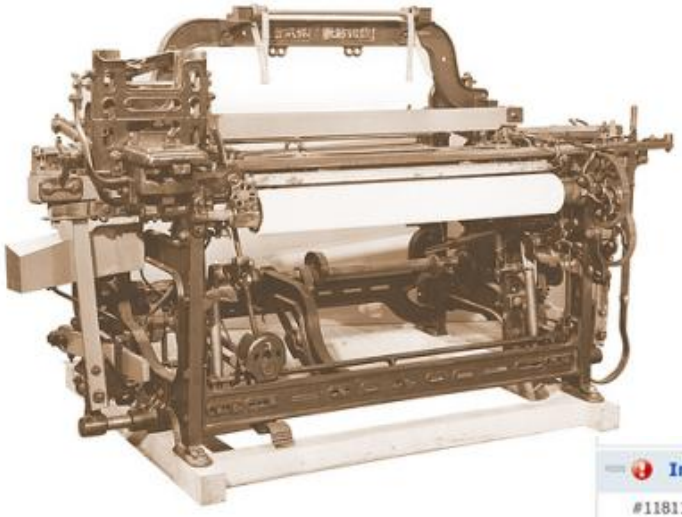
Quality

Quality

Defects per unit of valued work



Lean principle: Stop the Line



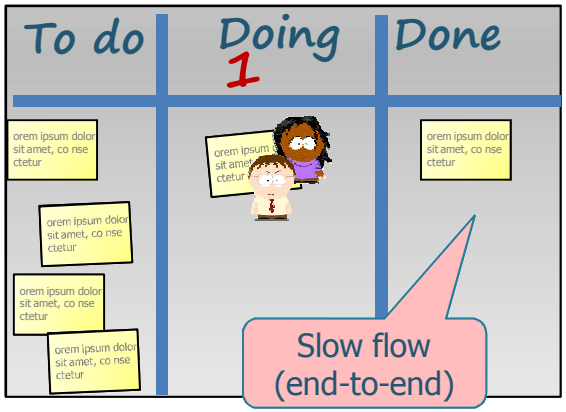
The build server is your friend! Really!

Henrik Kniberg

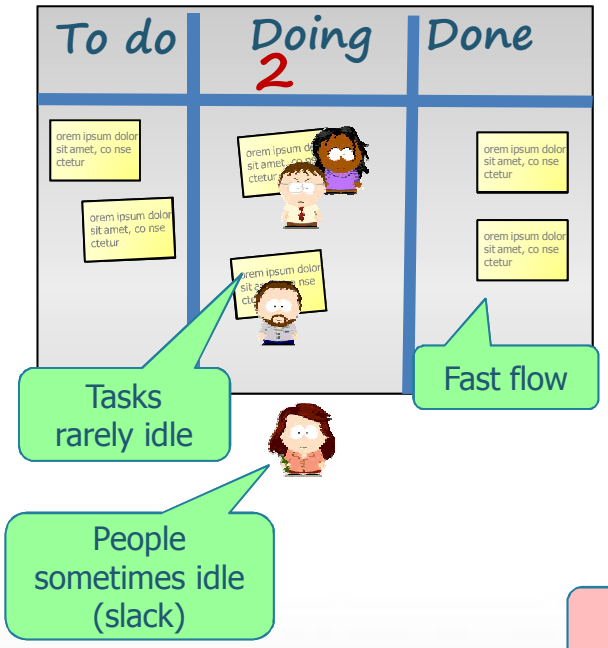
Optimizing the WIP limit

Optimizing the WIP limit

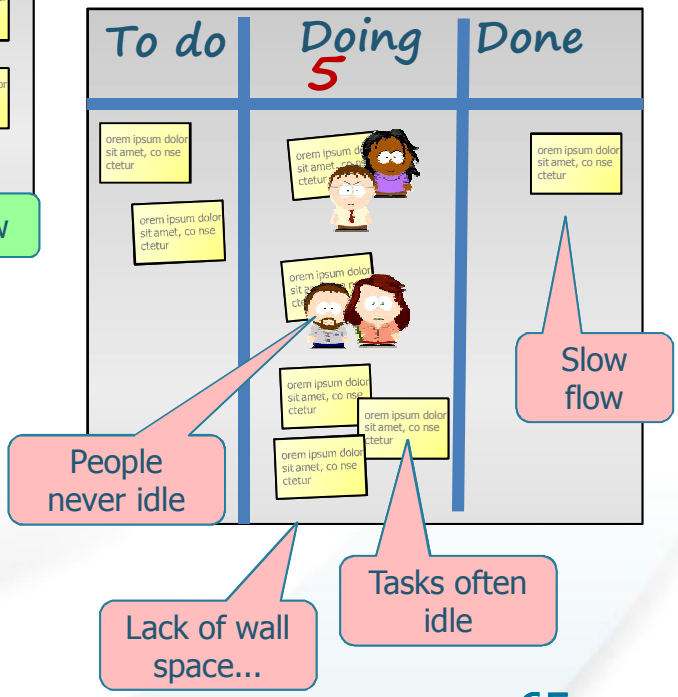
Too low WIP limit



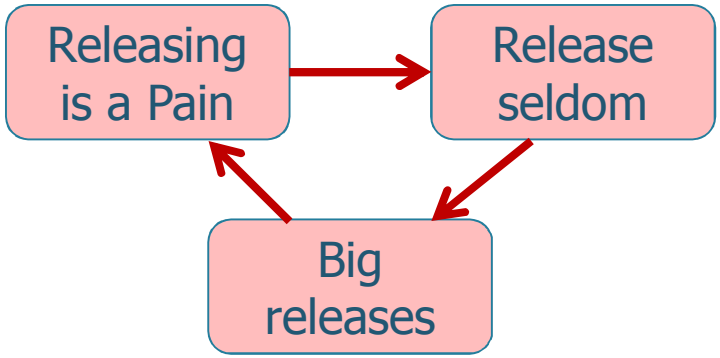
Just Right WIP limit

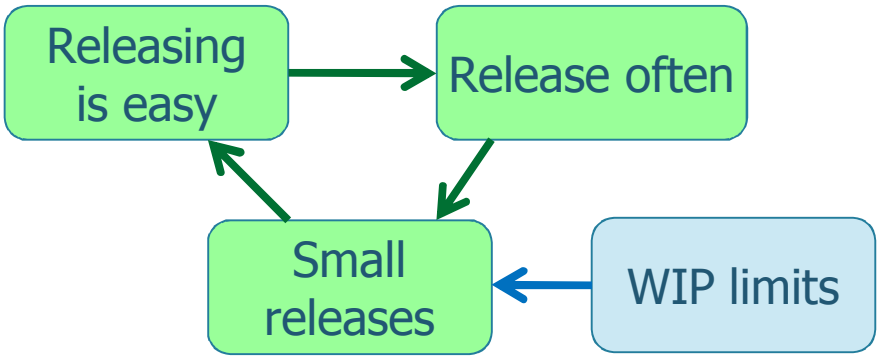


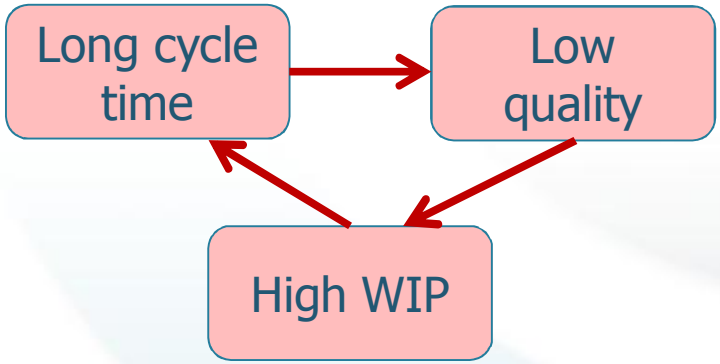
Too high WIP limit

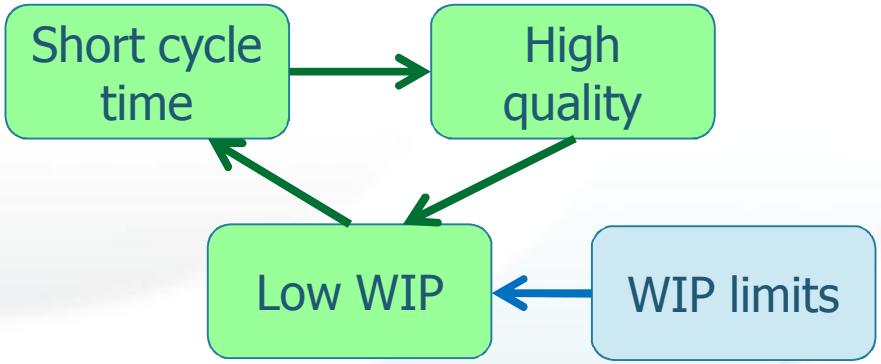


Use WIP limits to break vicious cycles









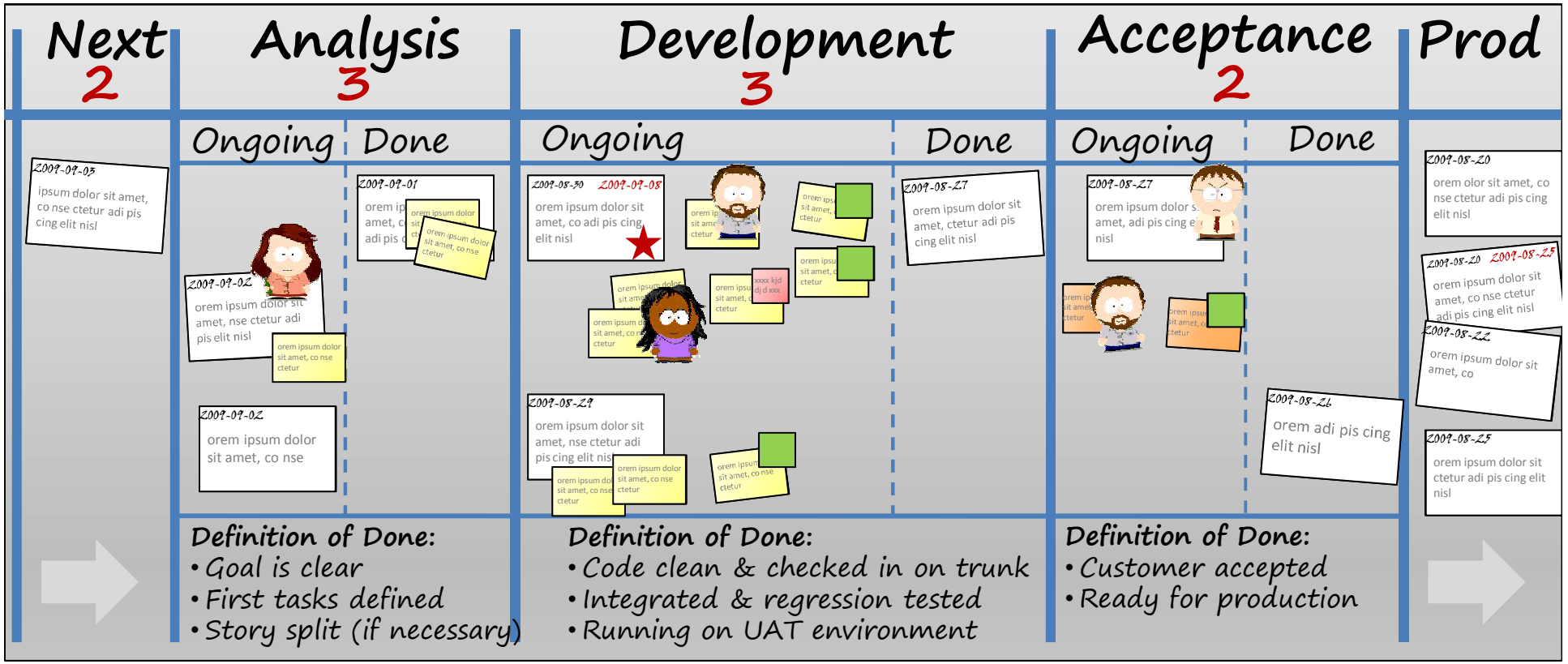
Evolution of a Kanban board

Henrik Kniberg

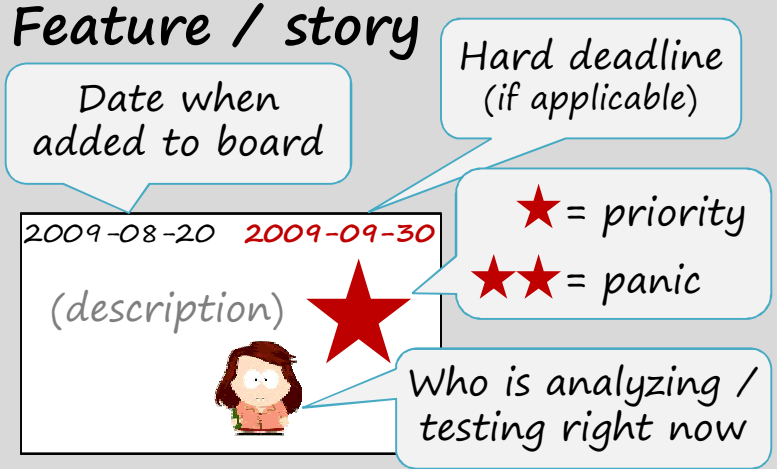
Kanban kick-start example

www.crisp.se/kanban/example

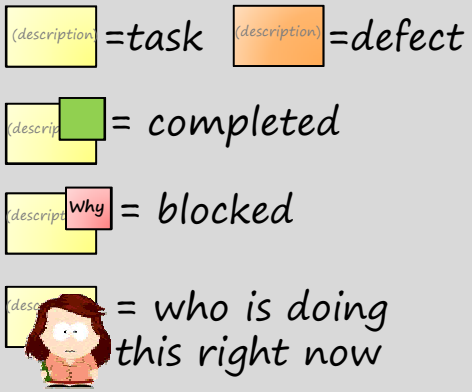
version 1.2
2009-11-16



Feature / story



Task / defect



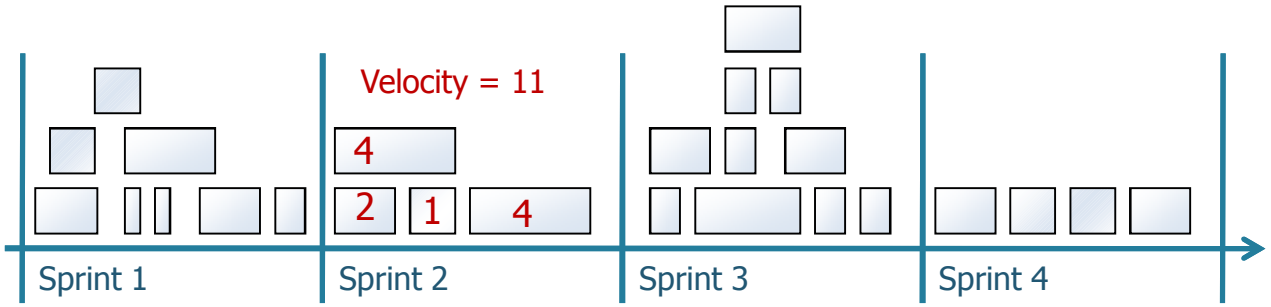
What to pull first

- **Panic**features ★★ (should be swarmed and kept moving. Interrupt other work and break WIP limits as necessary)
- **Priority** features ★
- Hard **deadline** features (only if deadline is at risk)
- **Oldest** features

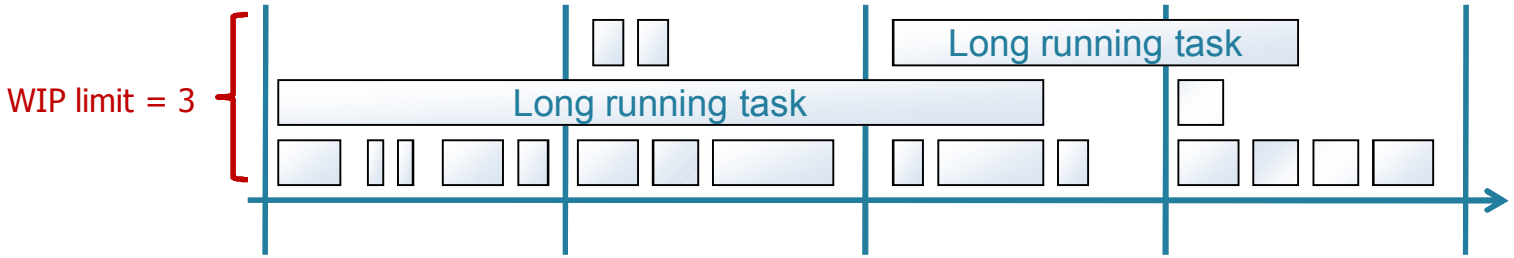
Estimation & planning

Estimating is optional in Kanban

Scrum

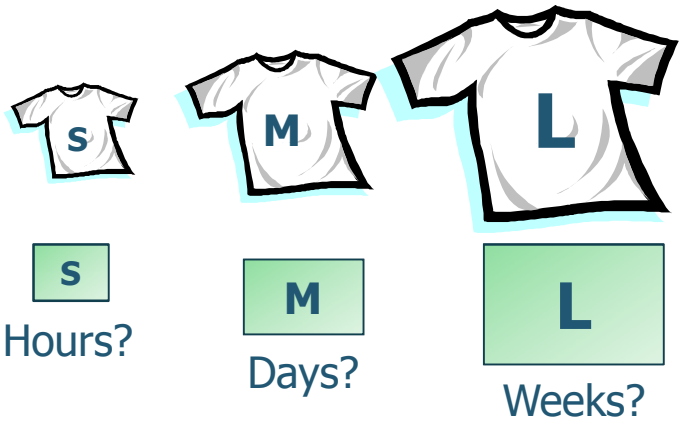


Kanban



Lightweight estimation techniques

T-shirt sizing



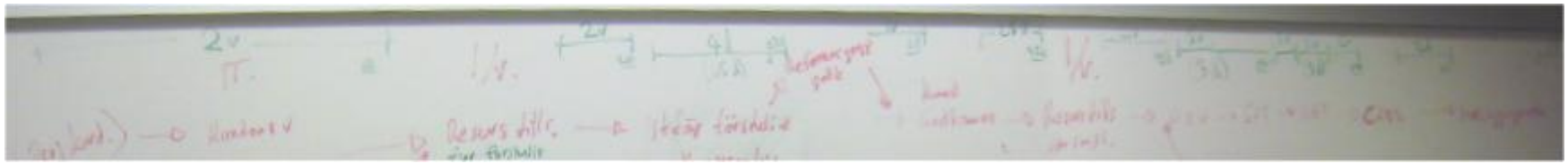
SLA – service level agreement

Example: SLA is 12 days

- Can we do this item within 12 days?
- If yes: just pull it in.
- If no: reject it, or break it down, or estimate/negotiate

Example: estimation paradox

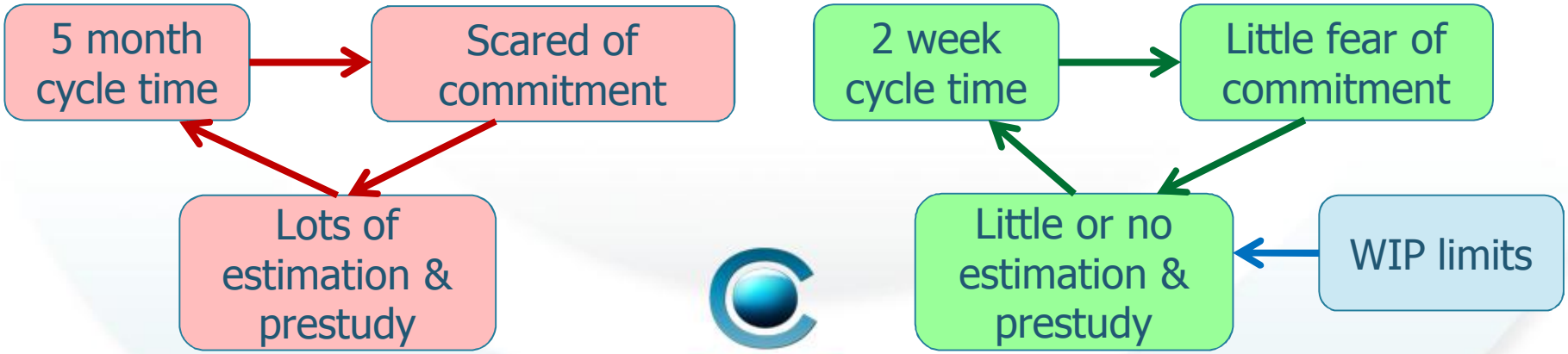
Estimation is sometimes used to manage a problem that was caused by the estimation itself



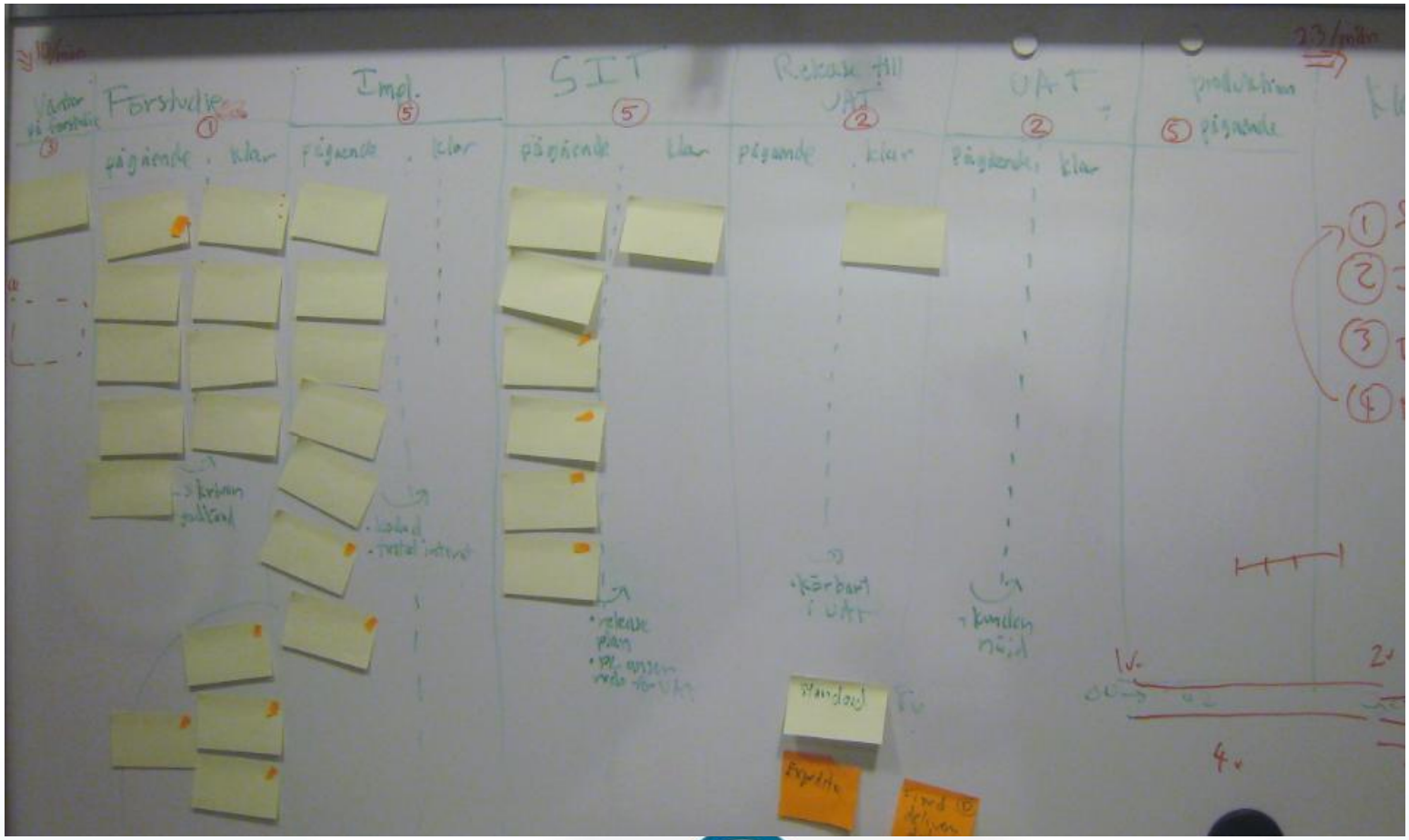
Prestudies, estimation, resource allocation, approval

Develop, test, deploy

$$\frac{\text{8 days work}}{\text{5 months cycle time}} = 9\ \%$$ Process cycle efficiency

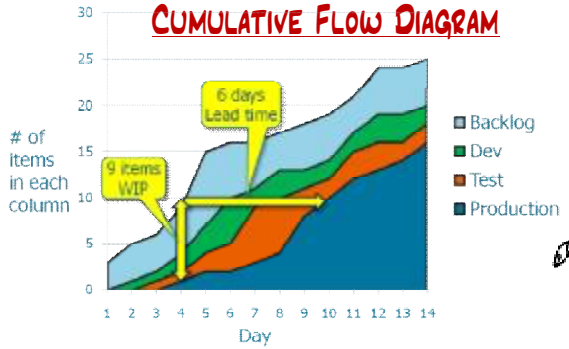


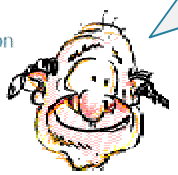
Using Kanban to visualize the problem



Metrics

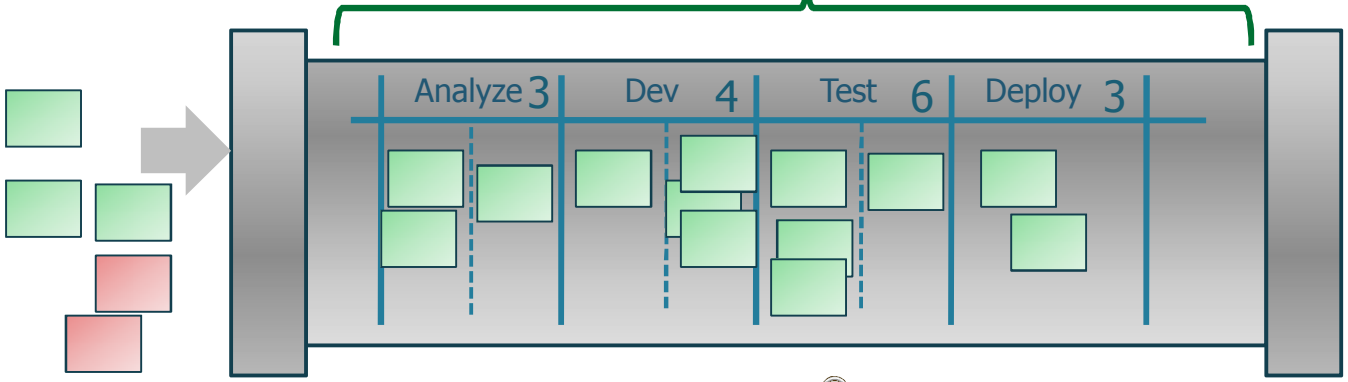
Metrics



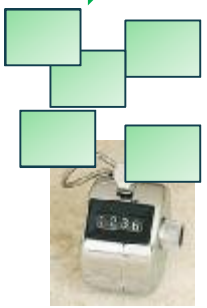


If we lower WIP limit from 6 to 2 in test we can probably halve the cycle time!

WIP (work in progress)



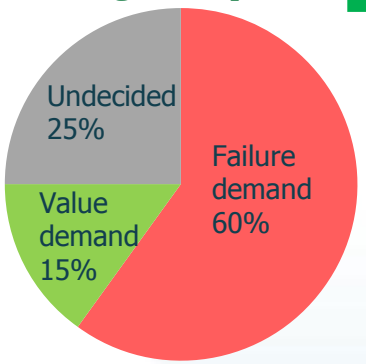
Release capacity (velocity)



Last week we released 8 things

Average capacity during Q1 was 6 story points per week

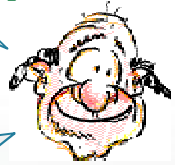
Quality



Cycle time

Item #25 took 7 days

Medium-sized MMFs take < 8 days 95% of the time.

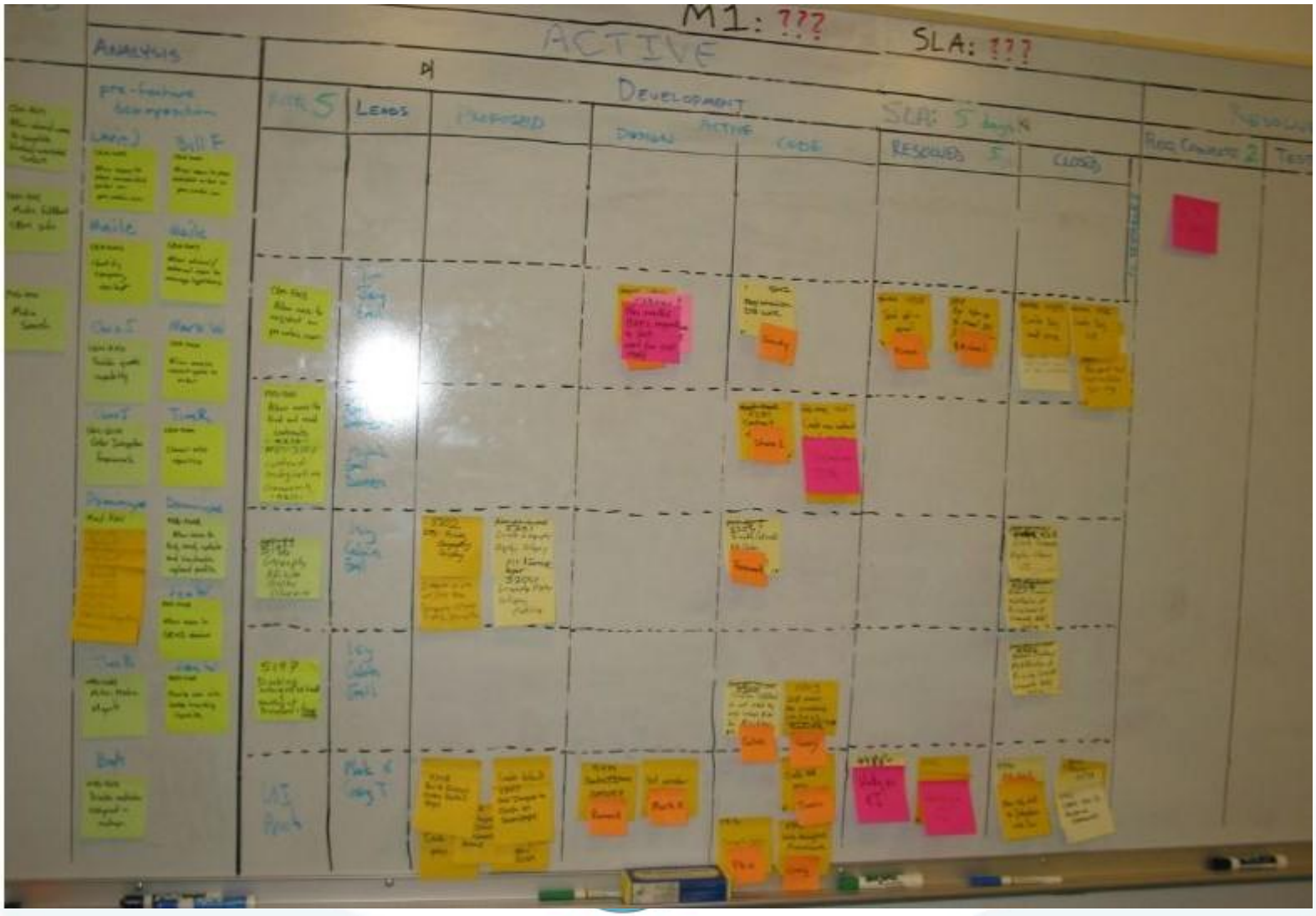


Misc patterns & Tips & Tricks

Process = collection of policies

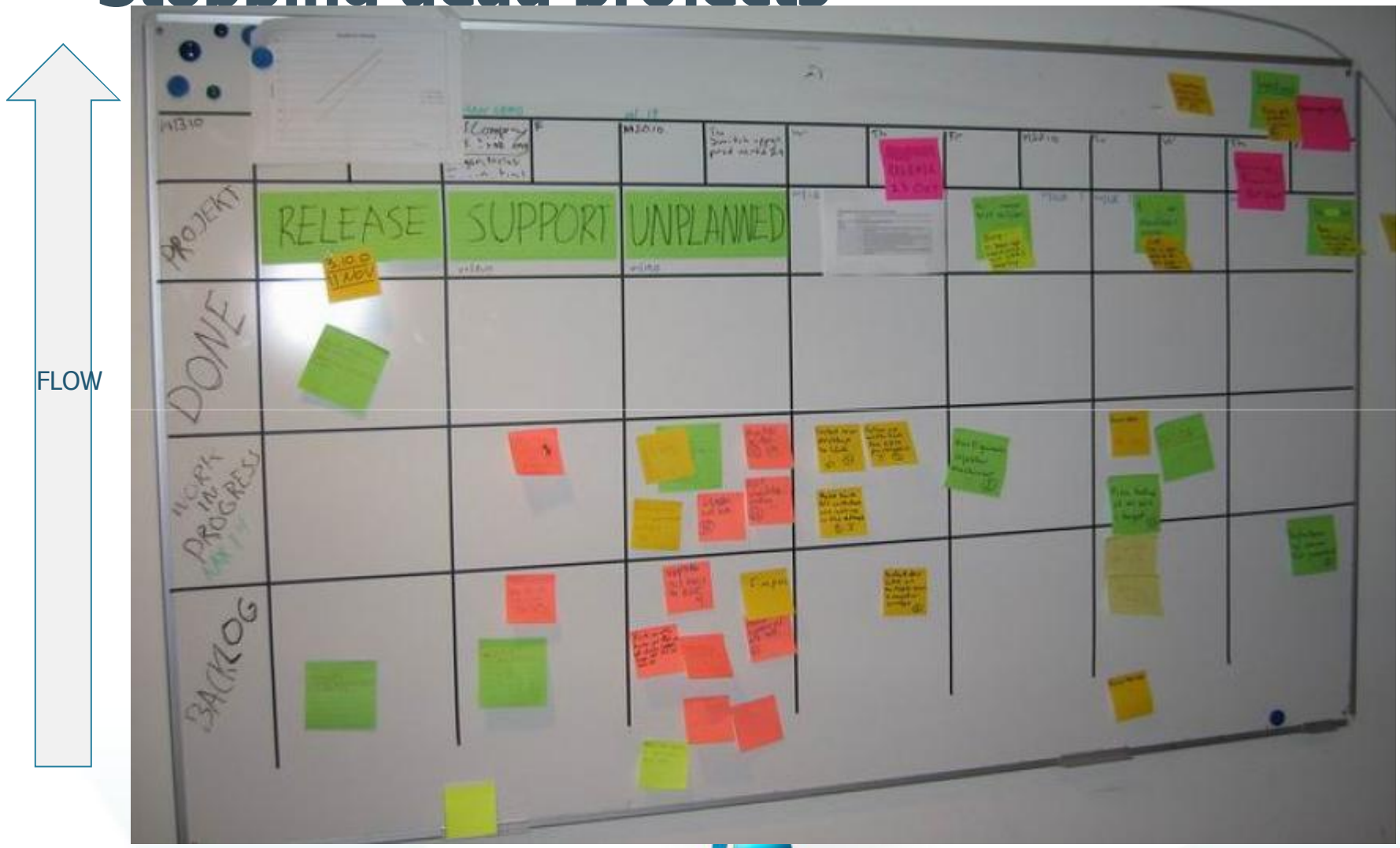
- **Visible on the board**
- **Brief**
- **Continuously renegotiated as needed**
- **Created on-demand**
- **Examples:**
  - Definition of Done
  - What to do when WIP limit is exceeded
  - Who decides priorities
  - When do we do planning?
  - Which types of work items to we pull first
  - Service level agreements

2 tier board with feature swimlanes



2 tier board-within-board

Stopping dead projects

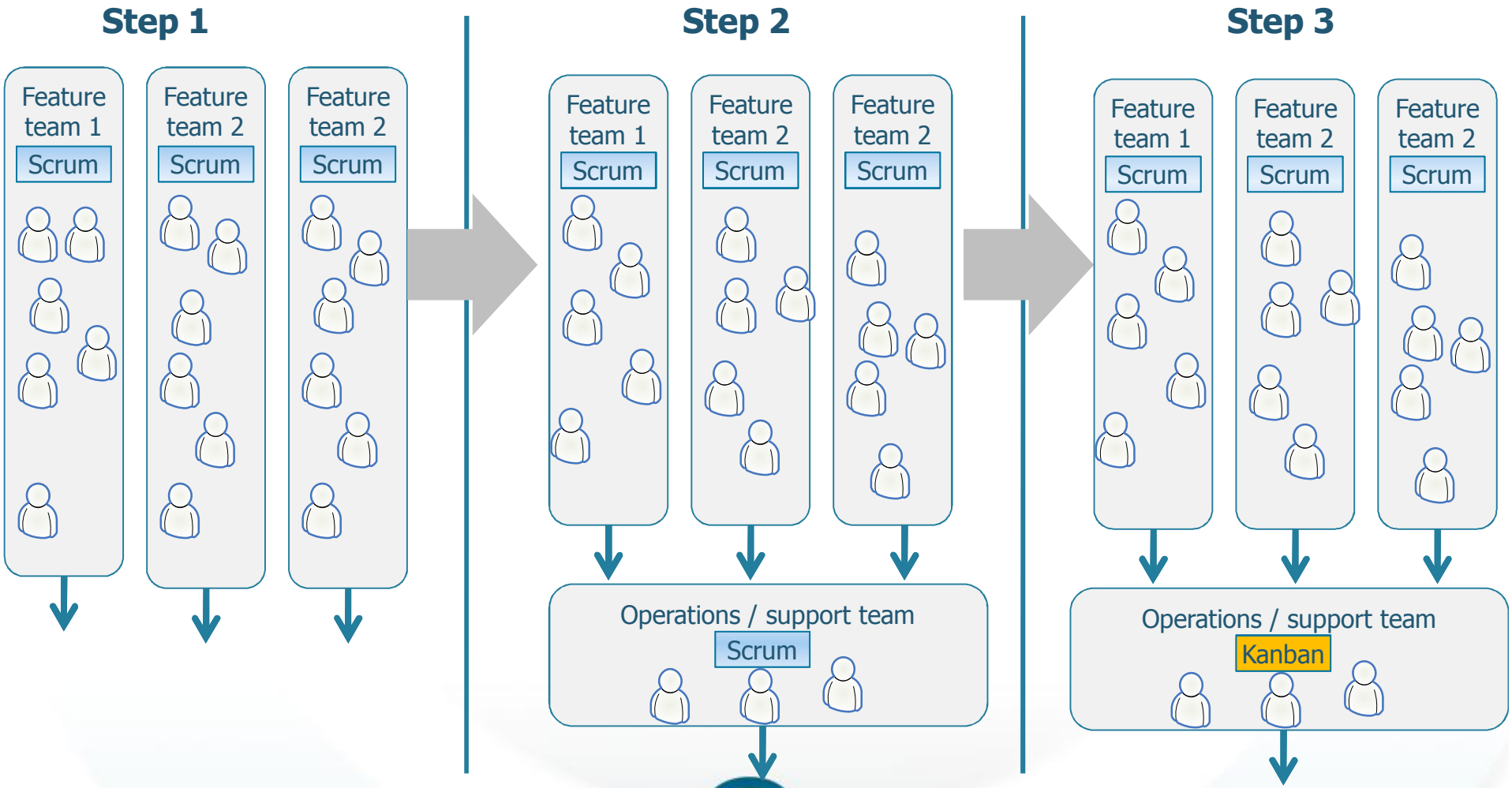


Manager removing impediments

- 2 slots on manager's door
- If both are full, team can add a new one if they remove a less important one
- Team decides when issue is solved

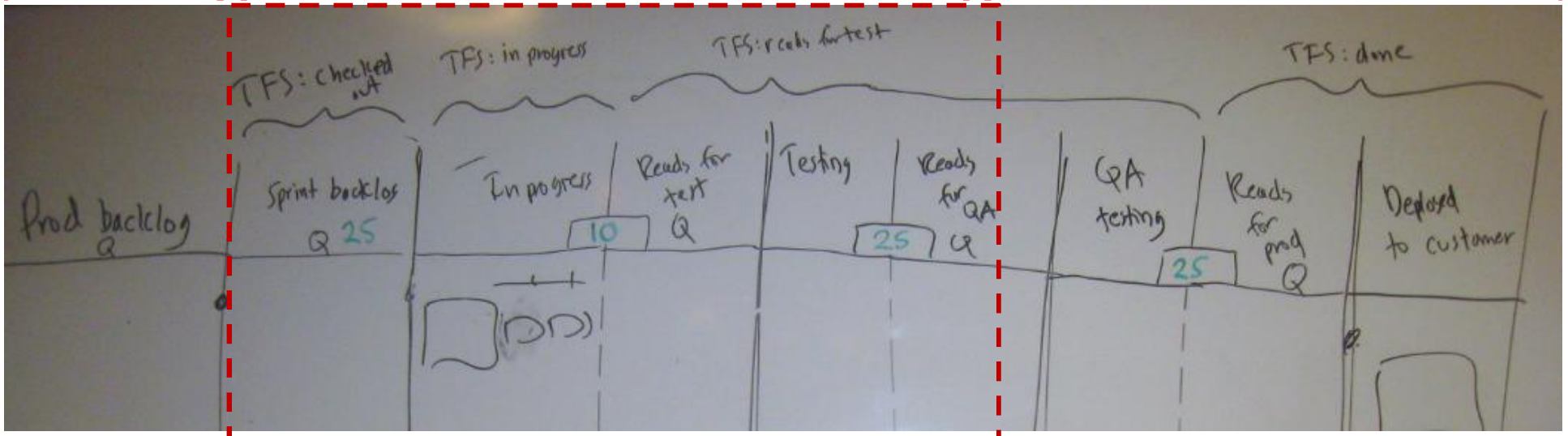
Bootstrapping Kanban

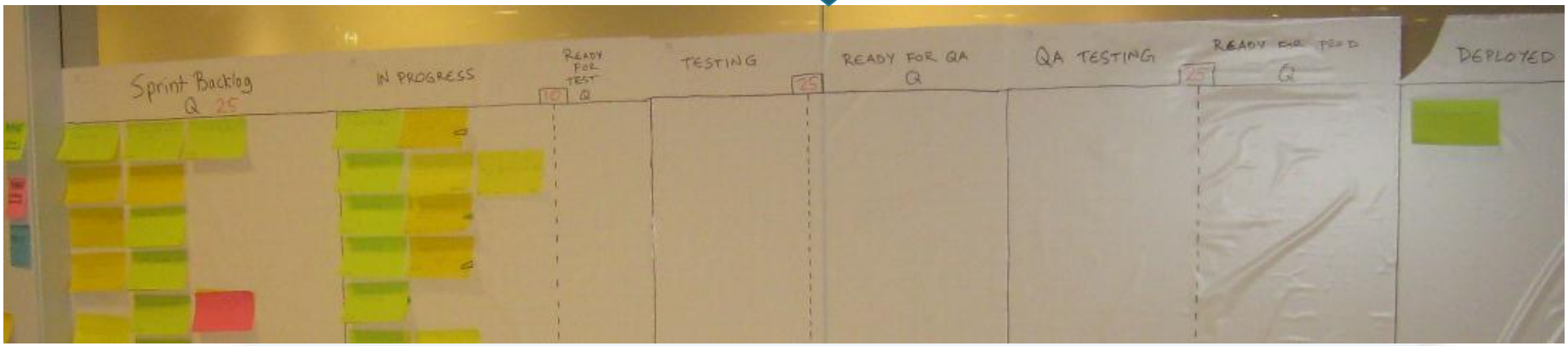
Typical Scrum => Kanban evolution

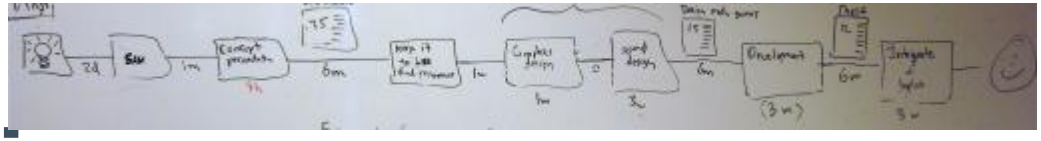


Example: Kanbanizing a Scrum team

Before sprint | Sprint | After sprint

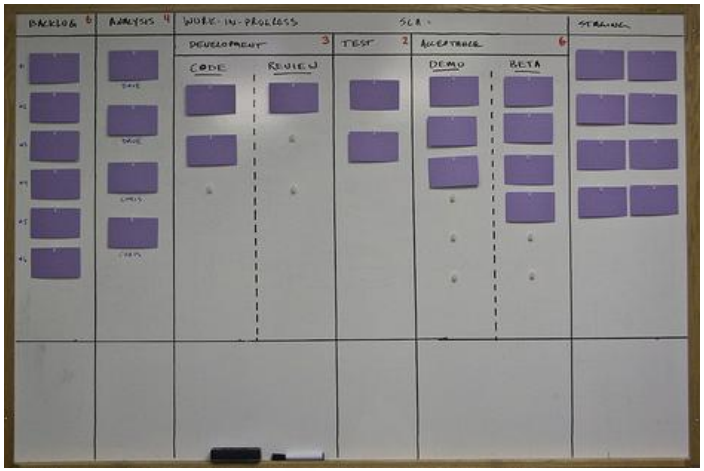






Kanban bootstrapping

1. **Define your value stream**
   - Which part do you intend to control?
   - Where is your input & output?
2. **Define work item types** Task Bug
3. **Meet with upstream & downstream stakeholders**
   - Policies, WIP limits
   - Input coordination
   - Output coordination
   - Target cycle time
     (SLA – service level agreement)
4. **Create Kanban board**
   (+ optionally electronic system)
5. **Agree on time for daily standup**
6. **Agree on when to do retrospectives / operations reviews**
7. **Go! Iteratively improve.**

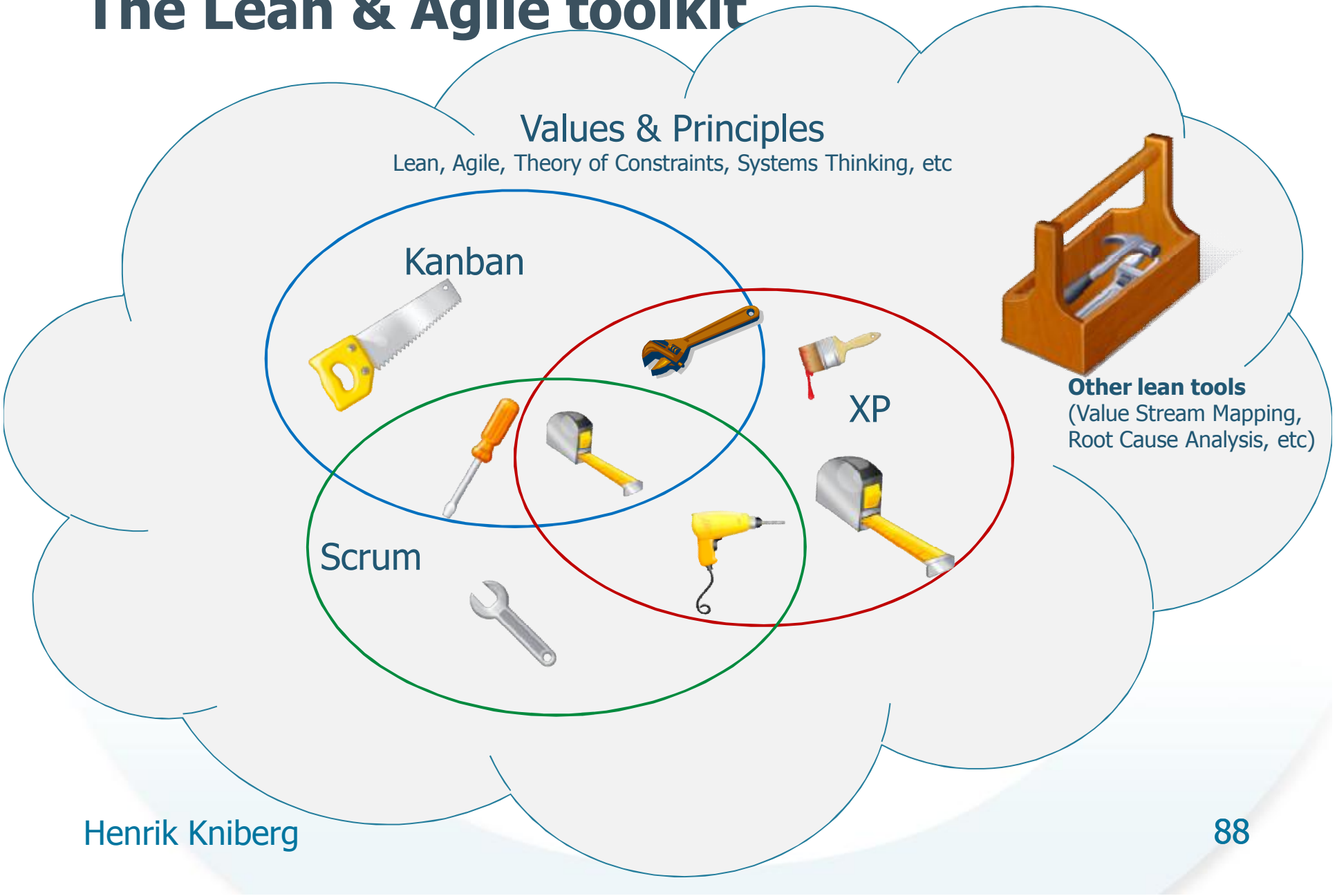


Wrapup

Recommendations

- **Visualize existing process first, *then* optimize.**
- **Put the board in the team room**
- **Focus on exposing problems**
- **Keep the board clean & simple** (just like your code...)
- **Don't try to get it perfect from start. Let it evolve.**
- **Don't overuse metrics**
- **Involve the team & managers & stakeholders**
- **Take photos**
- **Make policies & agreements explicit. Negotiate.**
- **Consider getting a coach for Kanban bootstrapping**

The Lean & Agile toolkit



Perfection is a direction, not a place

Expand your toolkit!

www.crisp.se/utbildning

Kanban starting points on the web:
http://www.crisp.se/kanban
http://www.limitedwipsociety.com

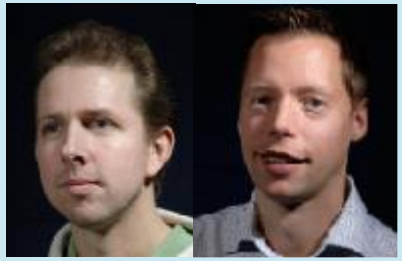
Leading Lean Software Development
March 4-5

Tom & Mary Poppendieck & Henrik Kniberg

Certified ScrumMaster & Kanban
Feb 17-19

Kanban Applied
Feb 23

Henrik Kniberg & Mattias Skarin



Kanban coaching workshop
April 29-31

David Anderson

Henrik Kniberg

crisp