

Experience Report

A “rough and ready” application of Lean and Kanban

BBC **iPlayer**

Katherine Kirk, 14 Oct 11

kkirk@rallydev.com

This presentation represents the personal views of the presenter and does not necessarily reflect the BBC

Note: I was not working at Rally during this project



- This is NOT
 - A slick presentation of answers
 - Full of metrics you can take back to the office
 - Any specific advice

- Story about overcoming edgecases
- Rough and ready experience report:
 - Emergency Driven Development

Assumptions

- You have basic understanding of Kanban, Scrum and XP

Reminder: Potential Differentiators

Scrum	Kanban
Focus on time-boxed iterations	Focus on continuous flow
Velocity (work done per iteration)	Cycle time (time to delivery)
Iteration commitment	Limiting Work in Progress (WIP)
Burn-down	Continuous Flow Diagram
Scope of work in iteration	Quality of service
Scales with teams of teams	Scales larger within team
3 roles, 3 documents, 3 meetings	No prescriptions



My approach

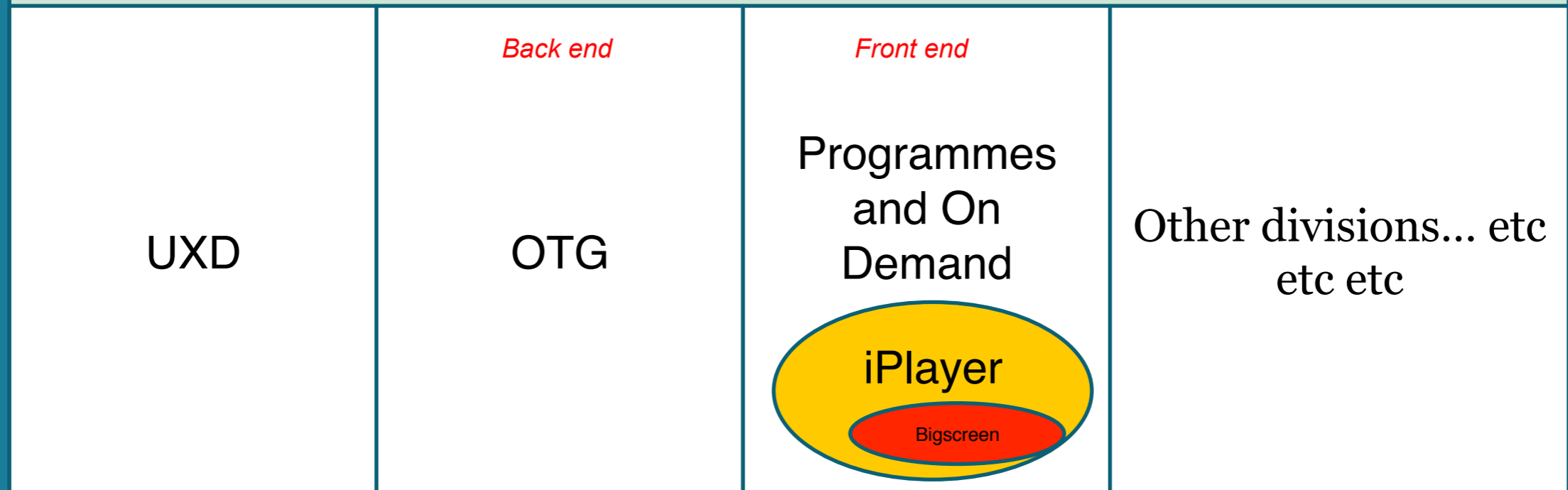
- Context
- Case study info
- Applying Lean and Kanban (short examples)
- Results
- Reflection

Context

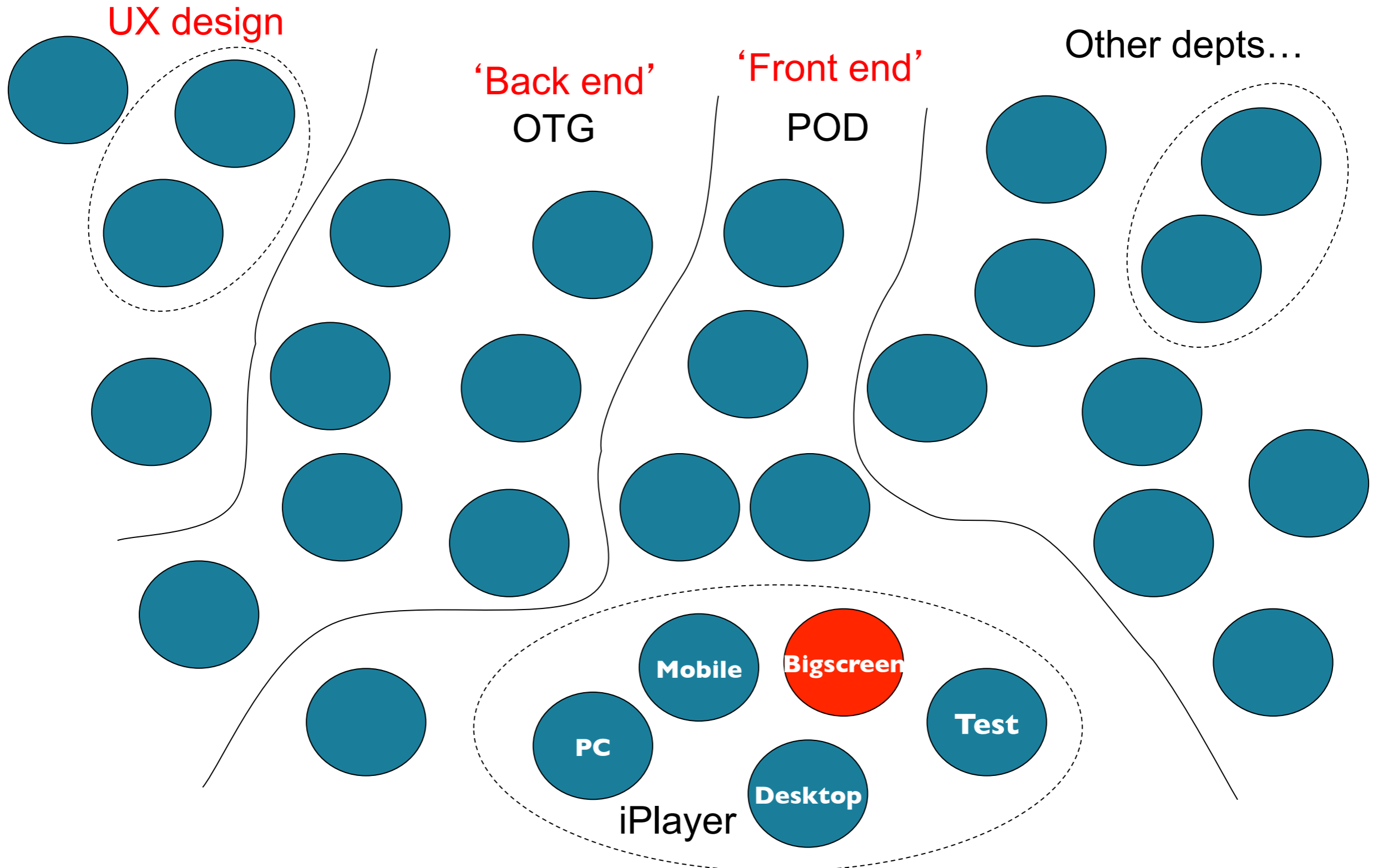
- 1 small team in a BIG organisation

Future Media and Technology (FM&T)

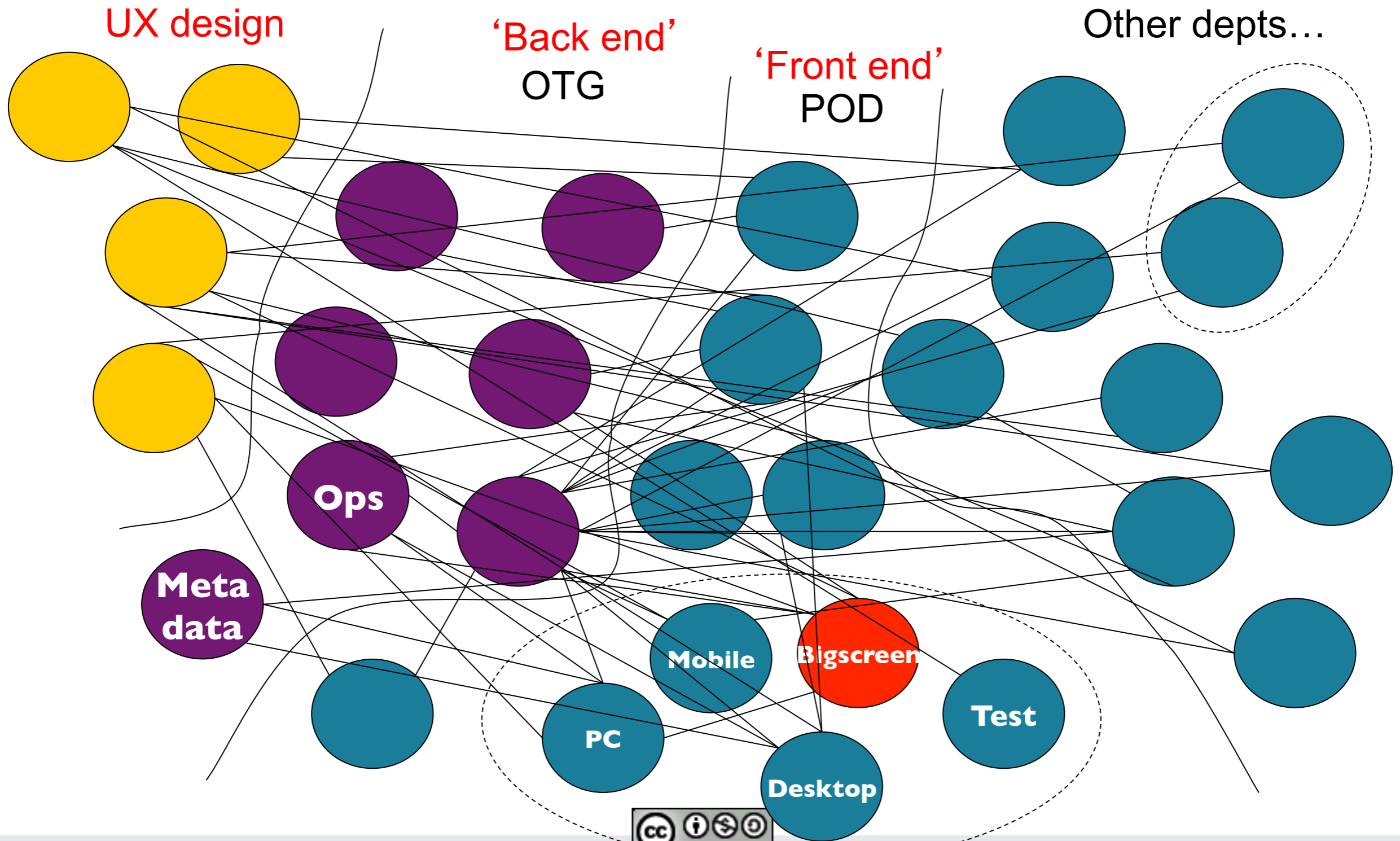
Online Media Group



Multiple teams in OMG, e.g....

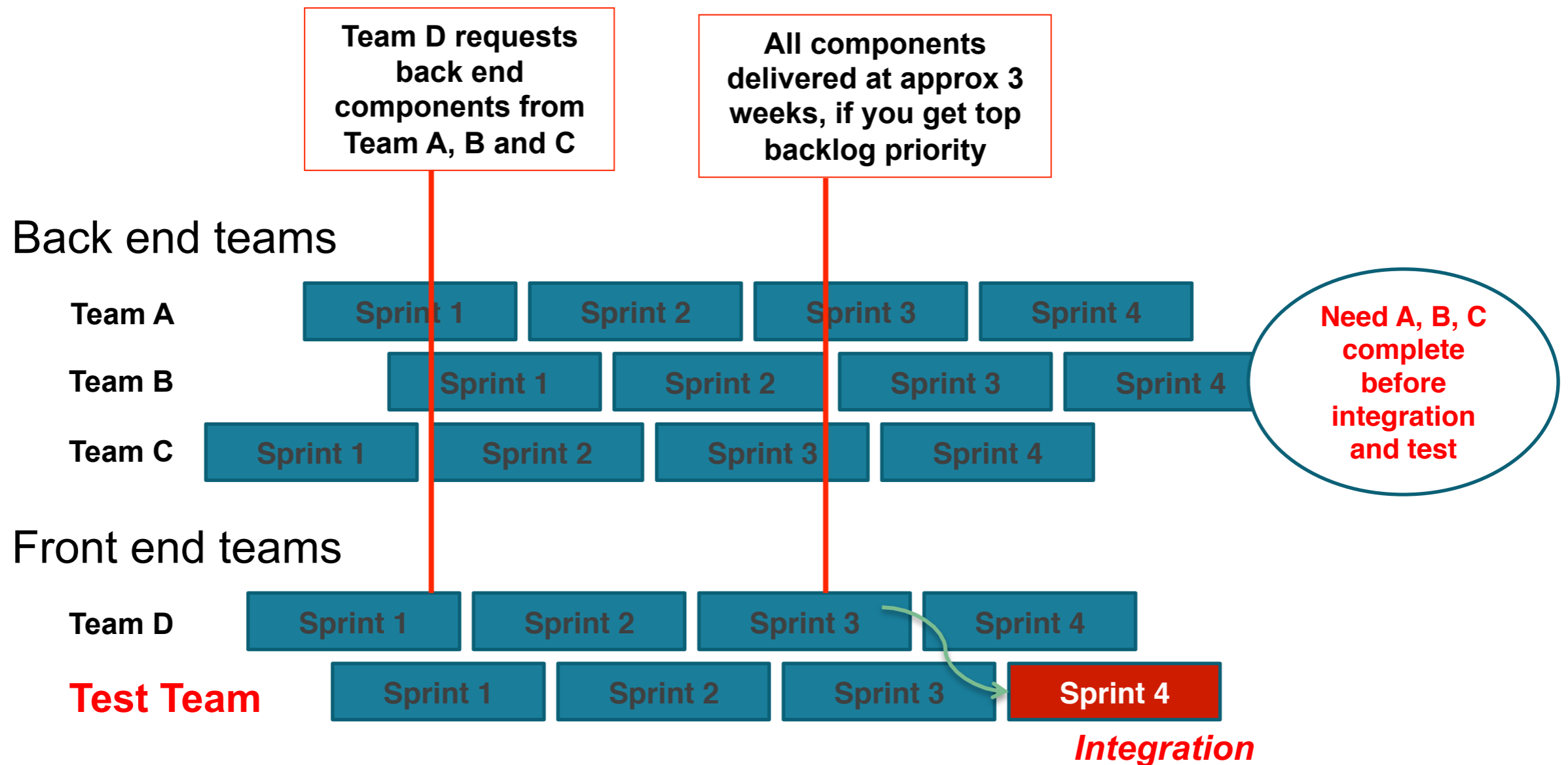


Competition for backlog priority



BIG org... Solve the Sprint Riddle

3 deliverables, 3 days worth of work = 5 weeks minimum delivery time
Minimum delay time: 1 sprint (2 weeks)



On top of actual dev and test work within PMs team

1 feature of main PC app

Above your team's workload

- 20 extra deliverables needed
- From 8 back-end and front-end teams
 - Across 3 divisions

1 full device app

Above your team's workload

- 30-40 extra deliverables needed
- From over 10 back-end and front-end teams
 - Across 4 divisions
- Multiple deliverables from 4 external companies

- Days of work = weeks of coordination
- Simple requests = overdose of online application forms
 - Building to dev environment
 - Changing or requesting a feeds

- High pressure
- Time critical
- High profile
- Highly complex system

The experience report

- March-September 2010
 - Needed rapid and successful response to a fast paced, very demanding live release schedule
- iPlayer
 - Online TV catchup service

iPlayer V2 customisation programme Bigscreen



Customising the iPlayer app for 20+ Bigscreen devices

Example: 1 top live device = 3.4 million users

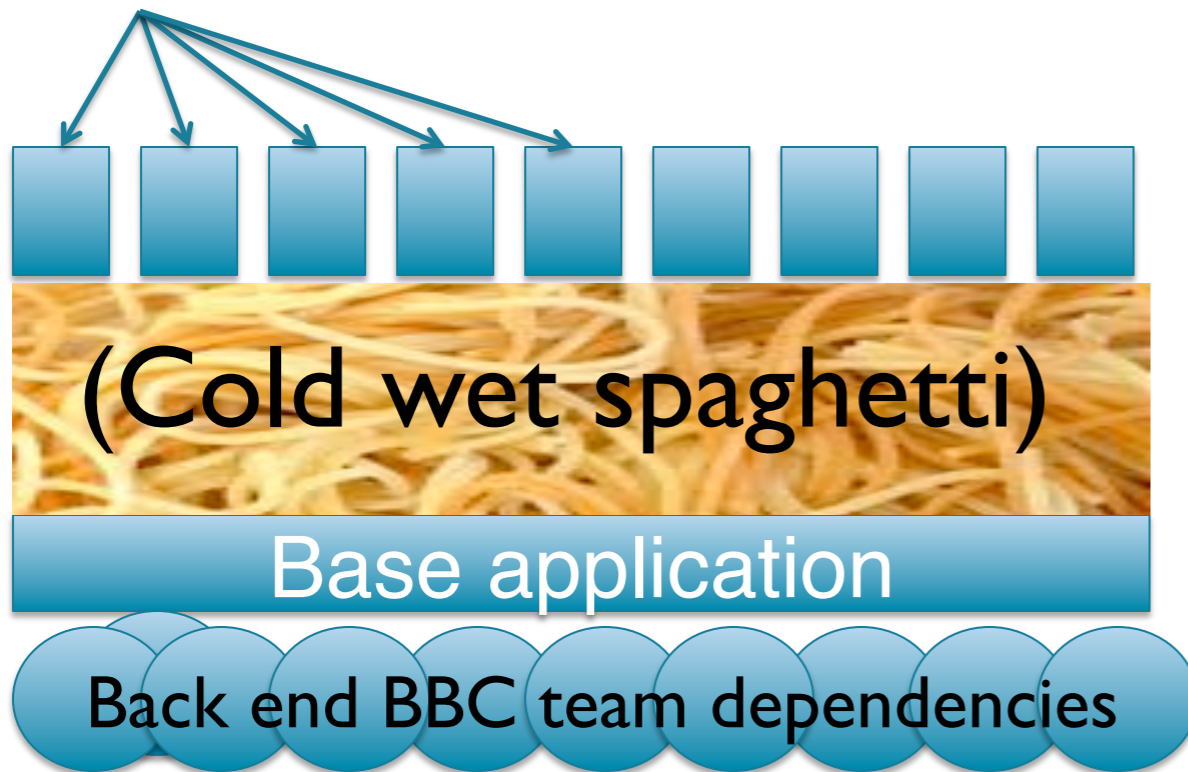
- General Manager's new dream
 - Team is realising his new ideas
 - Application has been built: start releasing!!!
 - Results expected ASAP
 - Live software must be delivered “now”
 - Stakeholder value must be realised “now”
 - Press and industry expectations are high
 - ‘Failure’ is unacceptable

- Small team of 2 dev, 1 tester, 1 delivery manager
- Working on ‘obscure/new’ technology
- Once application is built...
 - Average customisation = 3 months
- Multiple devices (around 20-30+) waiting for customisation
 - Deliver as many as possible

- Delivery Manager and star coder
 - Car crash management + over functioning
- Tester
 - Resentful and reminding everyone he's 'a perm and on loan' and 'no-ones his boss here'
- 2nd Dev
 - Bad case of second sibling syndrome
 - Underfunctioning

- Relatively ‘unknown’ technology
 - New technology and hardware
 - Browsers in devices at ‘1996 capability’
 - Trying to meet 2010 design
 - No specialists in the BBC or industry ‘yet’
- Competing for backlog priority against teams with multi-millions of users
- No processes
- No tracking system
- Part-time, main dev in Scotland, Tester ‘on loan

Vendor device customisations



vs

Team: **1 month contracts** (not rolling) except the Delivery Manager

- Who is our customer? Too new....
- 'No permission' to do TDD / refactoring etc
 - No time for planning or retrospectives
- Very reactive 'go live' – can't do sprints
 - must release 'on demand'
 - Fit in with vendor release plans
- No 'long term' strategy: **get it out now**
 - Expectation: React 'instantly' to industry / customer feedback
 - Backlog items changing continuously
- Dependency 'hell'
 - Sprint riddle across all dependencies in FM&T = reactive go live
 - Low priority in backlogs



So where do you start???

Do I give up on this project?

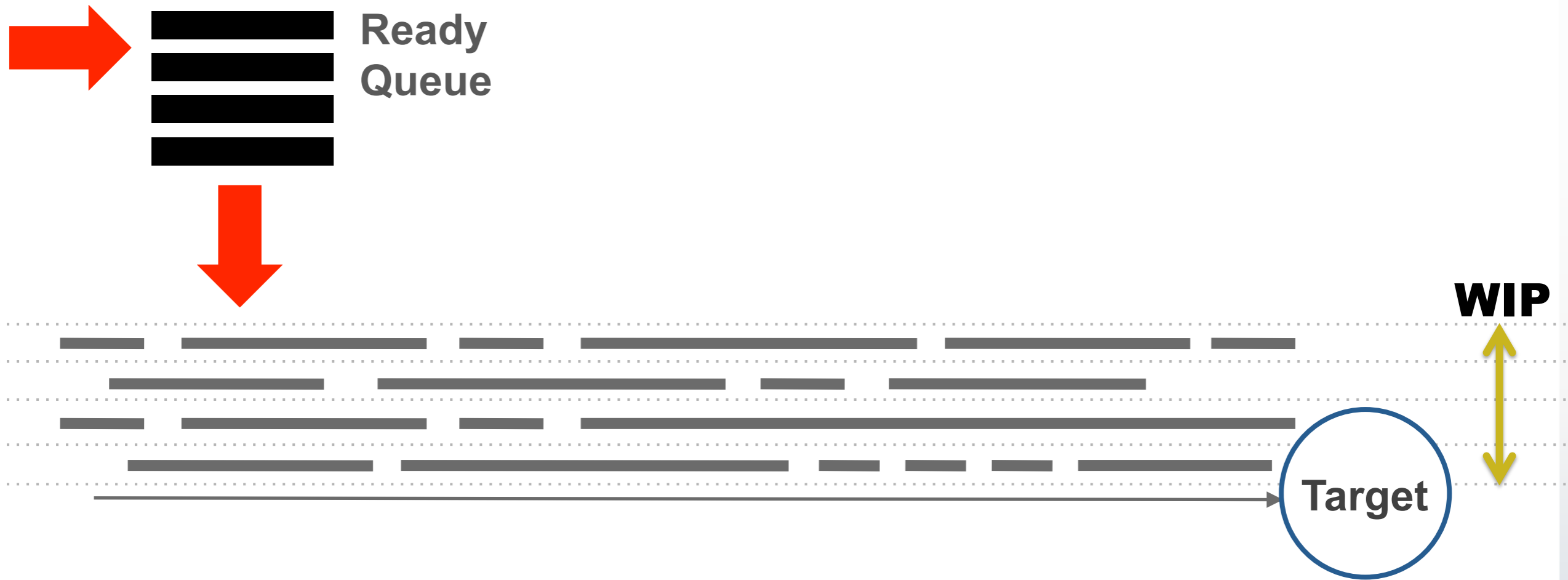
- Its cutting edge
- Great people

But...

- We couldn't use a text book method
- We couldn't solve everything
 - What could we use as a guide, to keep us on track?
- We couldn't change a lot at any given time
 - What could we do today?
 - What areas should we focus on ASAP?
 - When do we know we have delivered well?
 - How do we work around difficulties?

Kanban was promising

Scrum	Kanban
Planned work dominates	Demand work dominates
High ability to estimate	Low ability to estimate
Cross functional capabilities	Siloed capabilities
Self-contained	Unbounded outside dependencies
Low variation of work	High variation of work



(all independent cadences)

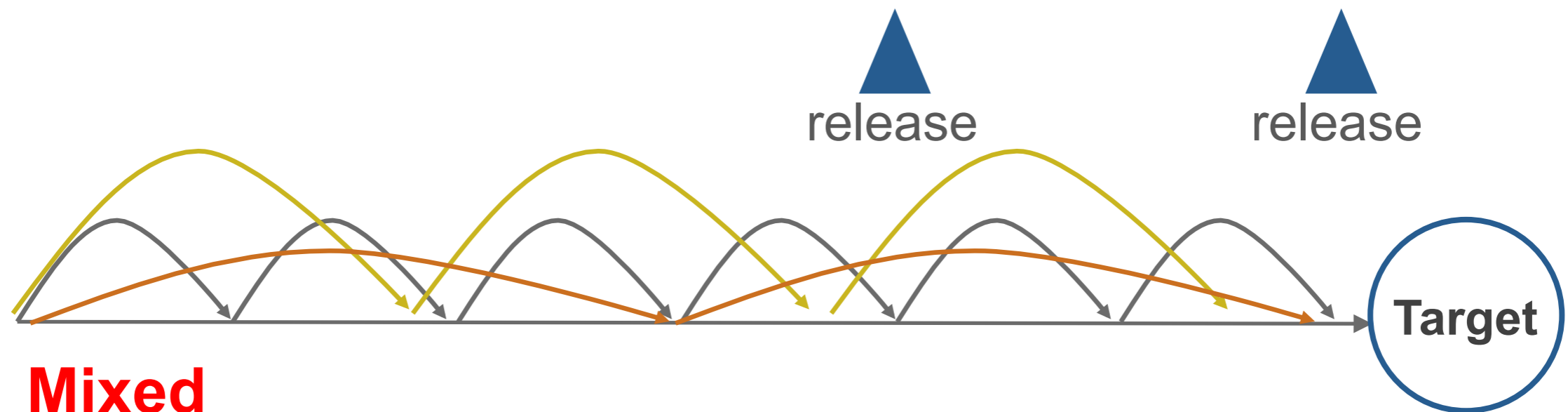

groom


prioritize

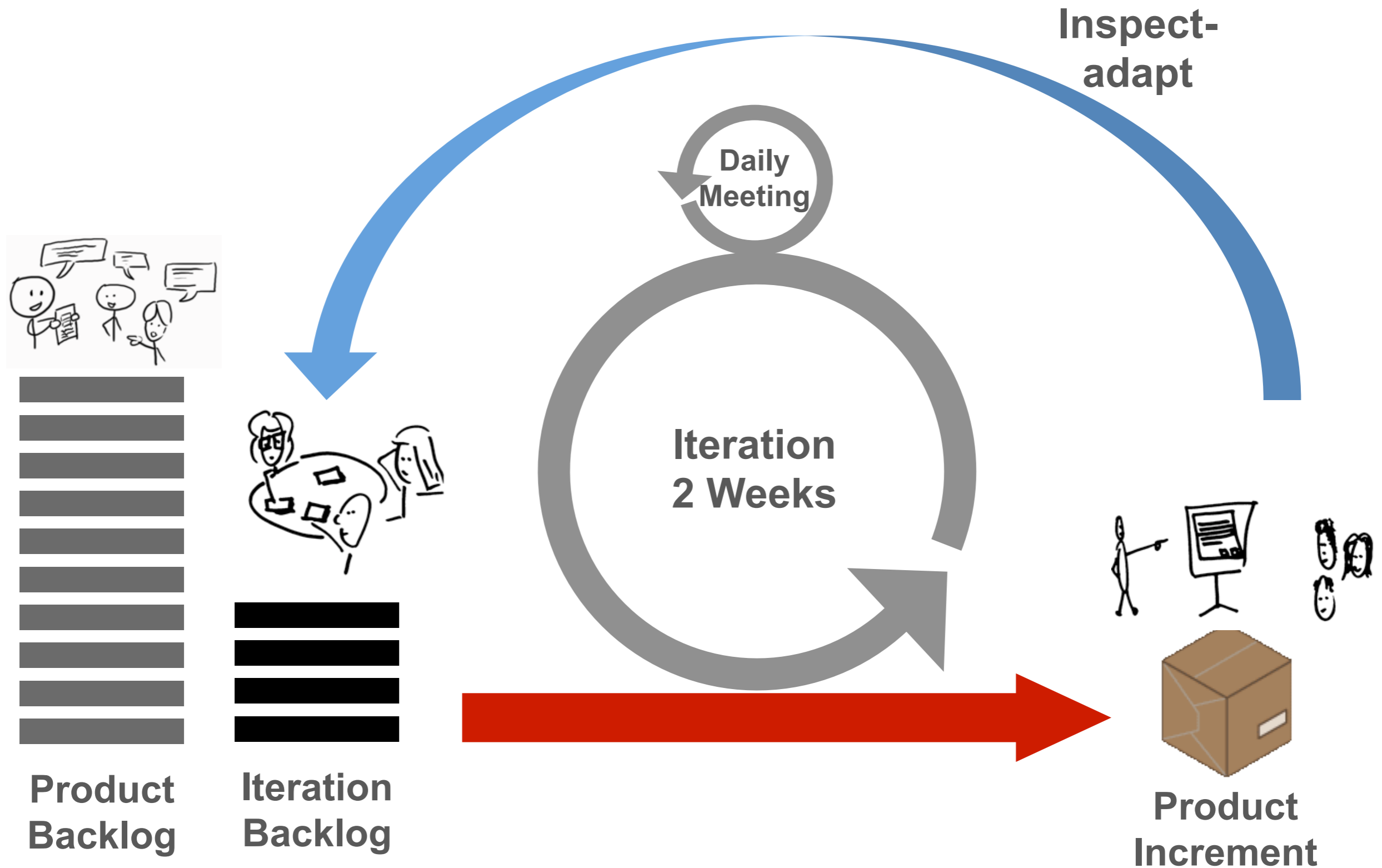

release


retrospect

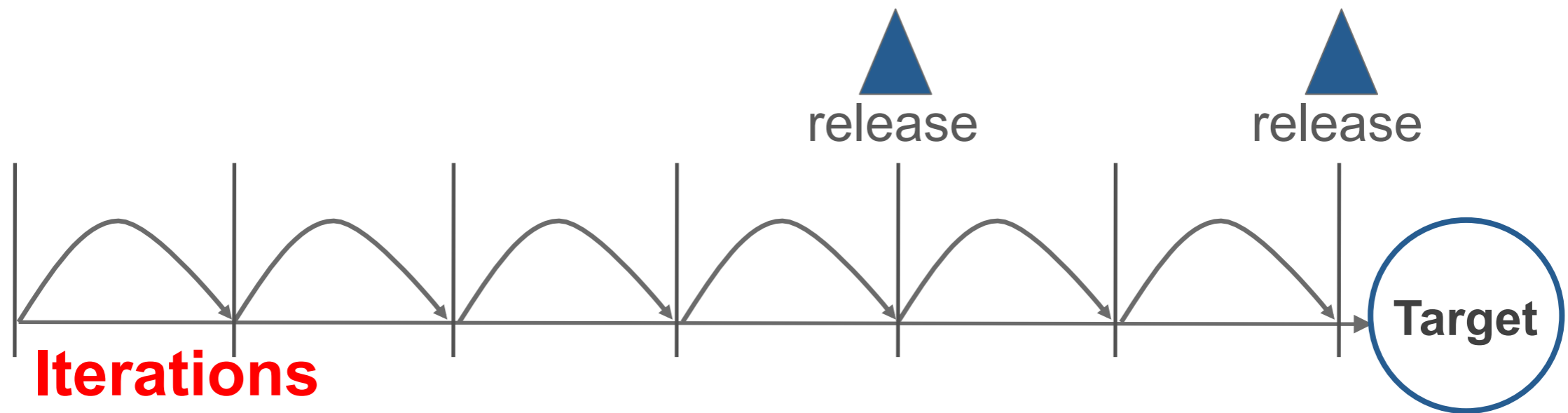

release



**Mixed
Cadences**



Scrum workflow



Leaning on Lean

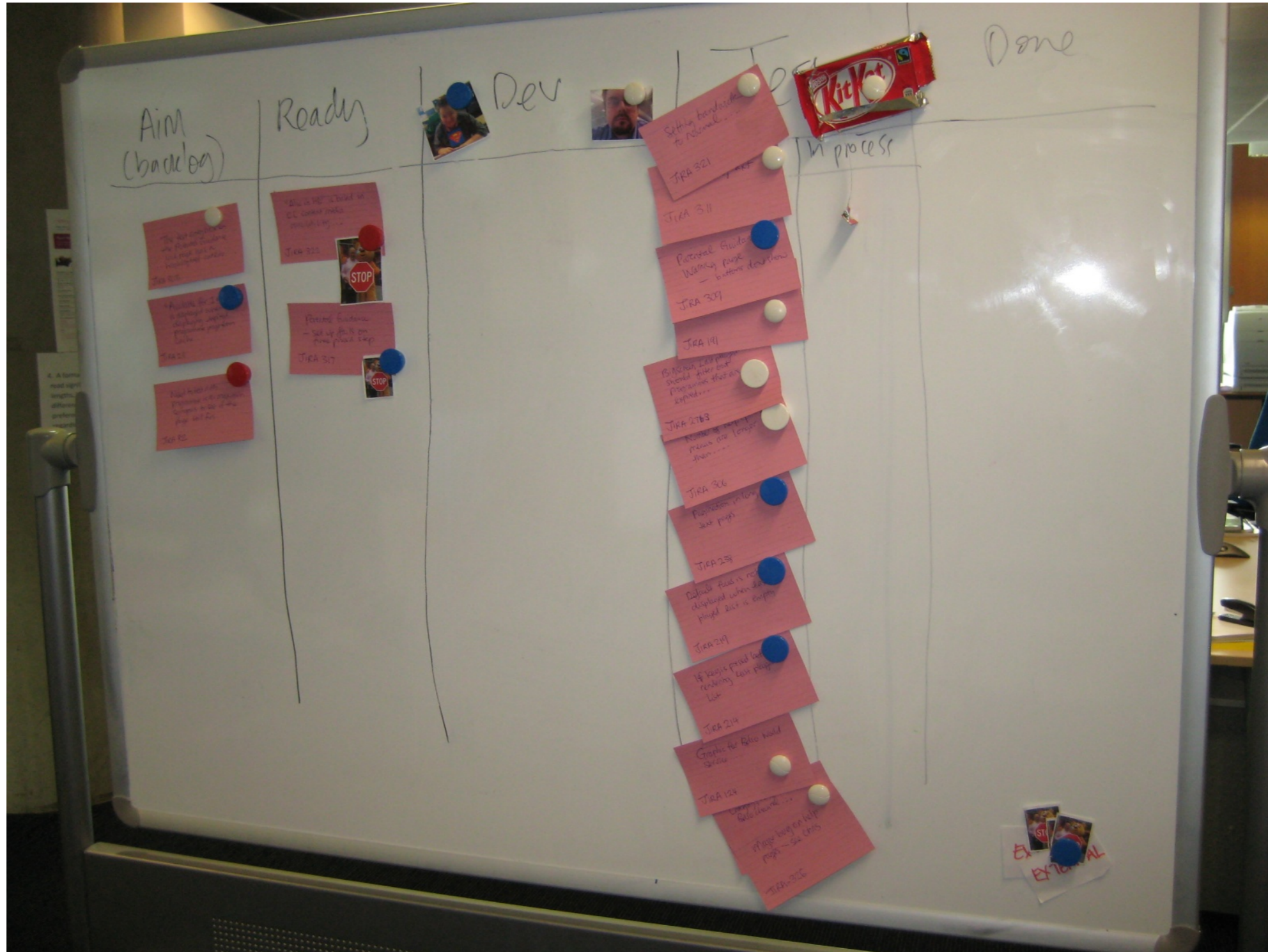
- Kanban and Lean seemed the only way...

- The first Kanban implementation for iPlayer

- Eliminate waste
- Amplify learning
- Decide as late as possible
- Deliver as fast as possible
- Empower the team
- Build integrity in
- See the whole

- Value stream mapping
 - Identified and established sources of waste
 - Collaborative
 - Utilised Kanban techniques

Started simple (we were under stress)



Behaviour

- Delivery Manager and star coder
 - Car crash management + over functioning
- Tester
 - Resentful and reminding everyone he's 'a perm and on loan' and 'no-ones his boss here'
- 2nd Dev
 - Bad case of second sibling syndrome
 - Underfunctioning

Why?

- Dependency hell
- Backlog priority changes rapidly
- None can step in

- Takes 3 days to build to dev
- None had shared basic knowledge of where to test

- Didn't know where to start!
- Didn't know what the others were doing

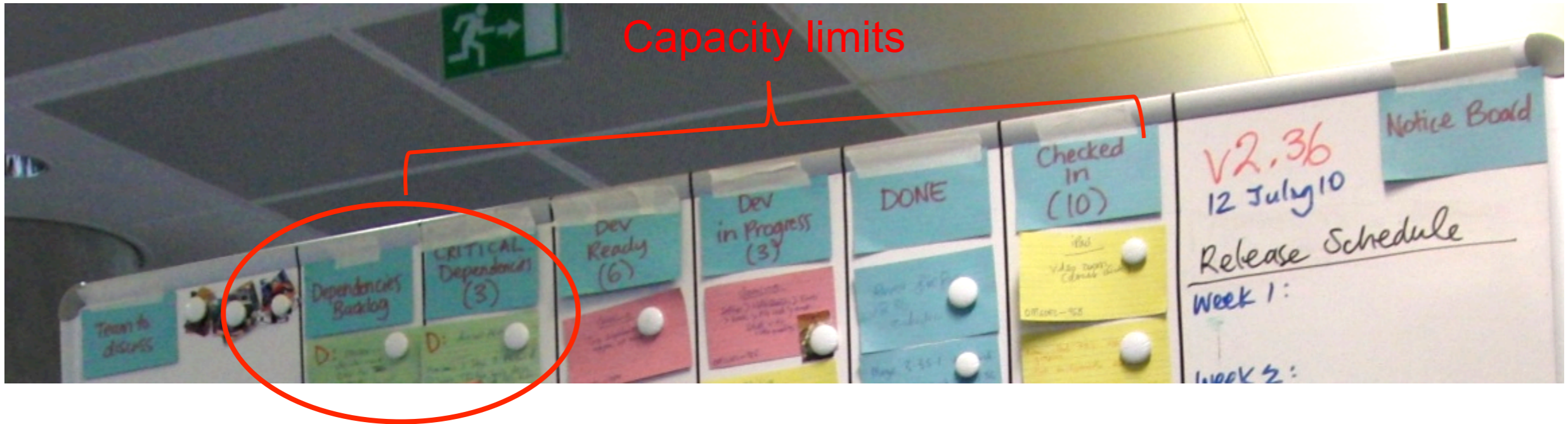
Suddenly we had a common mental model

- Talked about the workflow continuously
- Could see when someone was overloaded
 - Example: We all began helping to test
- Could see which blockers needed removal ASAP
 - Could identify upcoming issues earlier
- Came up with solutions / workarounds
 - Example: Donuts for Ops guys = speedier builds
- The backlog became EVERYONES backlog

- Things we could not change
 - Visualised / mitigated
 - Delays waiting for dependencies
 - Delays due to Scrum sprints
 - Management overhead

- Things we could change
 - Identified and actioned!
 - Be VERY quick to adapt
 - Rapid response to changes
 - Collaborate
 - Learn quickly

Primary aim... continuous flow



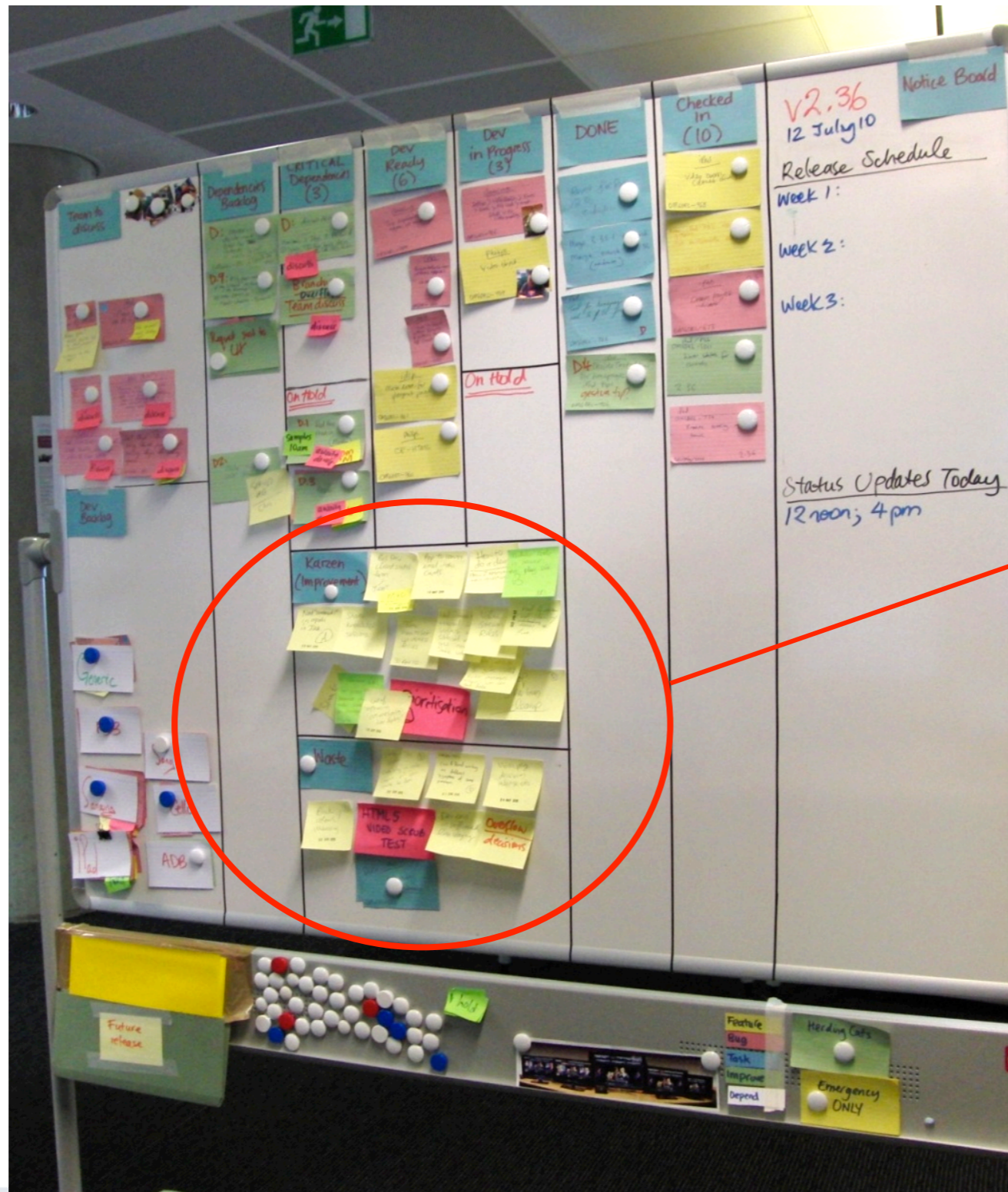
Dependencies (backlog)
CRITICAL dependencies
Blocker removal focus



**Smooth continuous
dev and test
workflow**

- We had to be inventive
 - Retrospectives?
 - Planning?

In place of retrospectives ...



Waste /
Improvement



Short,
focussed
solution
sessions

- Short 15 min chats
 - Driven by ‘demand’
 - Exceed waste ticket area capacity on board = action
 - What’s wasting our time?
 - Is there a pattern here?
 - What can we improve?
 - Are there any improvement suggestions which would solve a large chunk of the waste tickets?
 - What’s most important to us now?
 - Create ticket
 - Add to backlog

Planning as we go



Upcoming
workload



5 min
reviews, 3
times a day

- Solution
 - Delivery Manager reviews overall focus weekly
 - Picks top 3 devices to work on
 - Ensures dependency column 3 days ahead of dev ticket
 - Team collaboratively reviews focus 3 times a day
 - Adjusts backlog of ‘options’
 - **What we CAN do till dependency blocks clear?**

- Communicate / share skills
 - Stand up and share, anytime
- Shortest iteration cycles possible
 - Increased feedback cycle
 - Continuous review of twitter / blogs

- Options based approach to backlog
- Delay decisions
 - Which device to go out, depends on vendor readiness
 - Working on a number of devices at once
 - Whittles down as live date gets closer
 - Whichever vendor is ready at live date, goes live
- Prioritise LAST minute
 - **Release only fully complete and tested elements**

- Quicker delivery = quicker feedback = quicker response = better product
 - Increased our learning and communication
 - No rule book, no specialists, it was up to us to ‘become the experts’
- Interestingly, other dependent departments sometimes saw this as an irritant

Principle: Empower the team



- Project Manager – facilitation role

- Delivery Manager – blocker removal and customer / stakeholder ‘conduit’
 - Weekly strategy
 - Result required + ideal timeline

- Team
 - Self organised: Choose what is needed, when you need to
 - Divide and prioritise against strategy





Results

- Only measure: working software
- Hardly any time for reflection
 - Would have been good to have had some measures but....

- iPad v2 ‘quick and dirty’ release
 - Notified 4 weeks ahead to deliver iPlayer app
 - Gave us 3 weeks to ‘drop dead’ date: iPad release in UK
 - Concurrently keep other devices ‘ready to go’
- Hugely positive reception


- Product named one of the EU top 3 IPTV benchmarks in IFA 2010
- Business Development highly engaged
 - Product promoted and coverage increased
 - Now a ‘well known’ division of the BBC
- Released 9 Devices in 6 months
 - No known defects EVER released live
- Met and exceeded delivery release schedule
 - Happy General Manager (uses our products)

And we got organised at the same time: resulting value stream

PLANNING BOARD | **TASK BOARD** | CHART BOARD | RELEASED

Context: Show all * | View - New | My Issues

New card | View Version: V2.38 | Compact | Outline - Expand all | Collapse all



PM REVIEW - 1 issues (0 minutes)	MGR REVIEW - 18 issues (0 minutes)	ON HOLD (6) - 8 issues (0 minutes)	BACKLOG - 0 issues (0 minutes)	DEV READY (6) - 2 issues (0 minutes)	DEV IN PROGRESS (3) - 1 issues (0 minutes)	DEV COMPLETE - 0 issues (0 minutes)	CHECKED IN (10) - 9 issues (0 minutes)	READY FOR TEST - 17 issues (0 minutes)	TEST IN PROGRESS (6) - 3 issues (0 minutes)
<ul style="list-style-type: none">1424 Process for external	<ul style="list-style-type: none">1315 Set up an1316 Can you1356 Review LG68 The1215 Review1212 BusDev:1235	<ul style="list-style-type: none">1239 Test site1241 Add email1242 Simplify1243 Add in1318 Philips - can1310 Interface1414		<ul style="list-style-type: none">1442 Build to1443 Build to Integration	<ul style="list-style-type: none">1451 Philips shows		<ul style="list-style-type: none">1412 Most1413 Highlight1310 PS31314 Scrollbar1317 If focus is1395 LG: Browser1418	<ul style="list-style-type: none">1211 Anthony32 "Also in HD"1093 Create1213 On player1317 TEST:1313 Pink line1411	<ul style="list-style-type: none">1382 Build to1401 Dependency1437 Build toiDCP-dev:

- 1 year on, the team...
 - Expanded to over double the size
 - Became ‘self managing’
 - Experimented with a variety of different methods
 - Got very organised
 - Poured all their v2 learnings into creating a fantastic v3 app using BDD
 - Viewed as Leads across the BBC
 - Regularly shared learnings at conferences and BBC events
 - Always met (and sometimes exceeded) delivery deadlines whilst adapting to inevitable change

- Did so well – got taken over
 - Able to handover without politics
 - Assist new team to take app live

Reflection: Difficulties

- Principle: Build integrity in
 - No test automation / refactoring possible
 - No time to ‘build integrity’ across all dependent teams in BBC (small cog, big machine)
 - Optimised /mitigated as much as we could

- Mitigation
 - Short, multiple releases to live
 - Earlier awareness of dependencies
 - Keep all options open as long as possible
 - Focus on Quality of Delivery

- Continuous Irritation
 - Stress = shut down or push issues away
 - Changing view to a continuous improvement scenario is a big shift

- Embarrassment and feeling threatened
 - Exposing issues needs courage and sensitivity and TIMING
 - Keeping the blame culture at bay is hard

- Facing issues you can't do much about
 - Its a difficult attitude shift from helpless to mitigation
 - You need to learn to work around it and focus on what is going well

- Lean Principle: See the whole
 - ... In reality... 1 small team in a big pond...
 - Sometimes, awareness is all you can have
- Kanban workflow visualisation led to questioning why
- Questioning why led to understanding the whole
- We couldn't change it sometimes, but we could work around it
- Seeing the whole helped mitigate against risks

Reflection: Overall

- Were tools we used to help change our view
 - To work with what we had
 - Overcome what we could
- Faced with edgewise-hell and tough times
 - Triggered questioning and discussion
 - Helped transform from a ‘Can’t Culture’ to a ‘What Can We Do (with what we have) Culture’
 - We used it to empower us to create our own solutions



- Even without a nice clean, thought out adoption
 - In chaos we got results
- This approach
 - Focused attention on value
 - Process
 - Business
 - Technology
 - Kept us continually improving and experimenting
 - Ensured **lessons were learned and applied**
 - Was a foundation for sustainable improvement

- Kanban is a great tool
 - BUT.... Use it in context and with intelligence and sensitivity
- Kanban creates useful and informative views
 - BUT.... You might need to mitigate against backlash of what you find
- Kanban helped navigate difficult edge-case issues when things looked impossible....
 - BUT... you need to be prepared to change your view!!!

Don't be afraid!

- If faced with edgecases... Innovate not only with technology but also your process
 - Just focus on the principles
- Its up to you, your team



Thank you