Climbing out of a crisis-loop at the BBC

Katherine Kirk
Raf Gemmail

QCon London 2013

Session code: 7531

Introduction: the comfort page

- Katherine Kirk, Independent
  - Was PM on this project
    - Background
      - Contracting for over 10 years
        - » Investment banks, Media companies, Trading companies… mostly large corporations
        - » Previously:
          - Rally Coach – John Deere, Philips, Continental, Petris etc
          - BBC - R&D, iPlayer, Core services
      - MSc Software Engineering, Oxford
- Raf Gemmail
  - Was Dev on this project

- One scenario
- Two perspectives

Disclaimer

- This is the view of the presenters NOT the BBC
  - The current team is working well

Keeping buzz words to a minimum

... swimlanes, policies, WIP limits, empowerment, cooperation, etc etc ...

- Instead:
  - Case study + plain language
- Why?
  - At the end of the day: its about getting stuff done

This pres is about

- Working past the industry sell
  - Do Scrum or Kanban ‘right’
- What happens if you can’t do Scrum or Kanban ‘properly’?
- Can you still be Agile/Lean
- Can you get out of a pretty bad crisis?

- We think we did

Format

- What was the crisis?
- What Scrum and Kanban we did ‘roughly’?
- What did we differently?
- Why did the crisis loop stop?

- Not a typical agile team scenario
  - Purely back end team
  - Not cross-functional
  - All Perl/Java devs doing same thing
  - No front end
  - No vertical slicing

In 3 months

- Calmed the crisis-to-crisis cycle that had been running for nearly 2 years
- Began building new solution
- Kept things running AND improved the process at the same time
- Turned around stakeholder relationships

- Despite
  - People leaving and a restructure

But we did everything ‘incorrectly’

Kanban-ish

Scrum-ish

So what did we do differently?

And were we still Agile/Lean if we didn’t follow the ‘rule book’?

Key factor in our ‘success’

- Agile/Lean are principles NOT methods
- This means you can use your brain to solve stuff, as long as it aligns with the principles(!)
- Hmmm....

THE CASE STUDY: CONTEXT

Team

- Specialist, metadata delivery back end team
- Create feeds to display content
  - Main ‘client’: iPlayer
  - Daily traffic peak of between 200 and 500 requests/second (Not including cached responses)
  - Over 700 playback formats
  - Servicing hundreds of devices
    - Mobile, IPTV, PC, tablets (in all variants and models)

Put into perspective

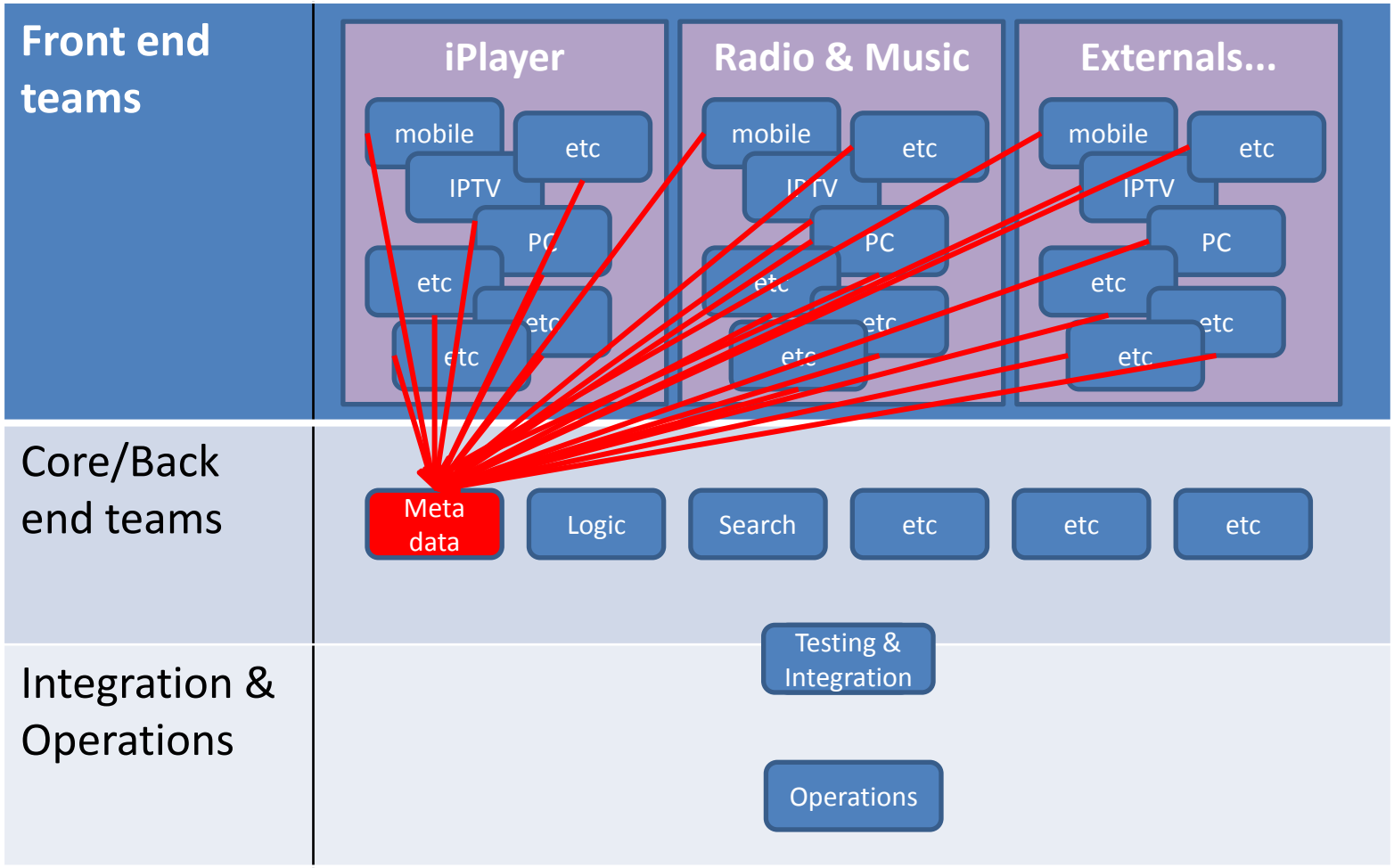
- "... 30m requests for iPlayer content via mobile or tablet in July [2012] alone
- [represents only] 20% of all requests for iPlayer programmes across all platforms... "

- Approx 150 million requests per month

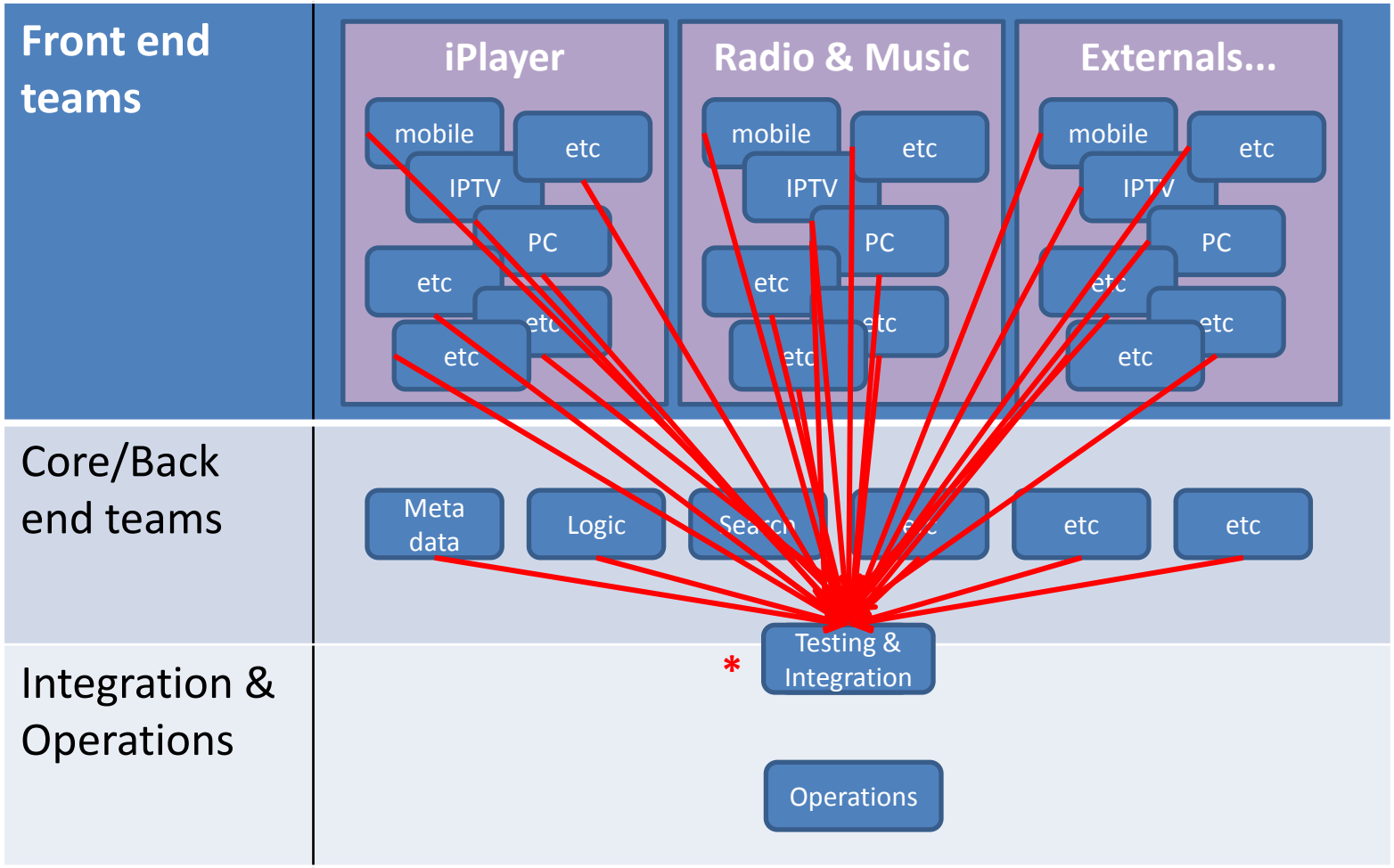
- No metadata feed = no content display
  - Front end teams are dependent
    - cannot display content without getting feed
    - cannot change or edit a feed – needs specialist expertise

http://www.bbc.co.uk/blogs/internet/posts/iplayer_mobile_downloads

Fierce backlog competition

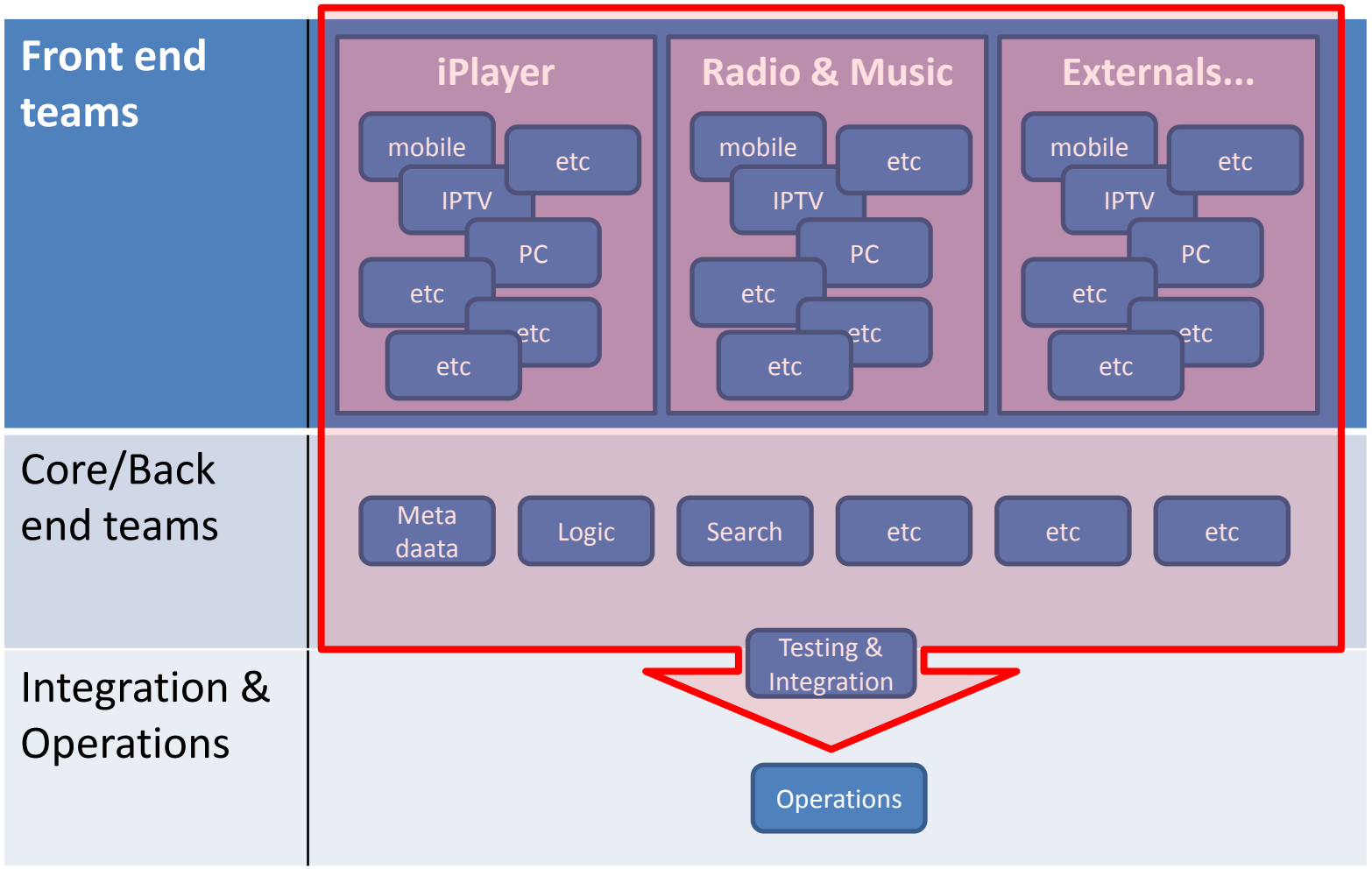


Integration & Test= 4 weeks min

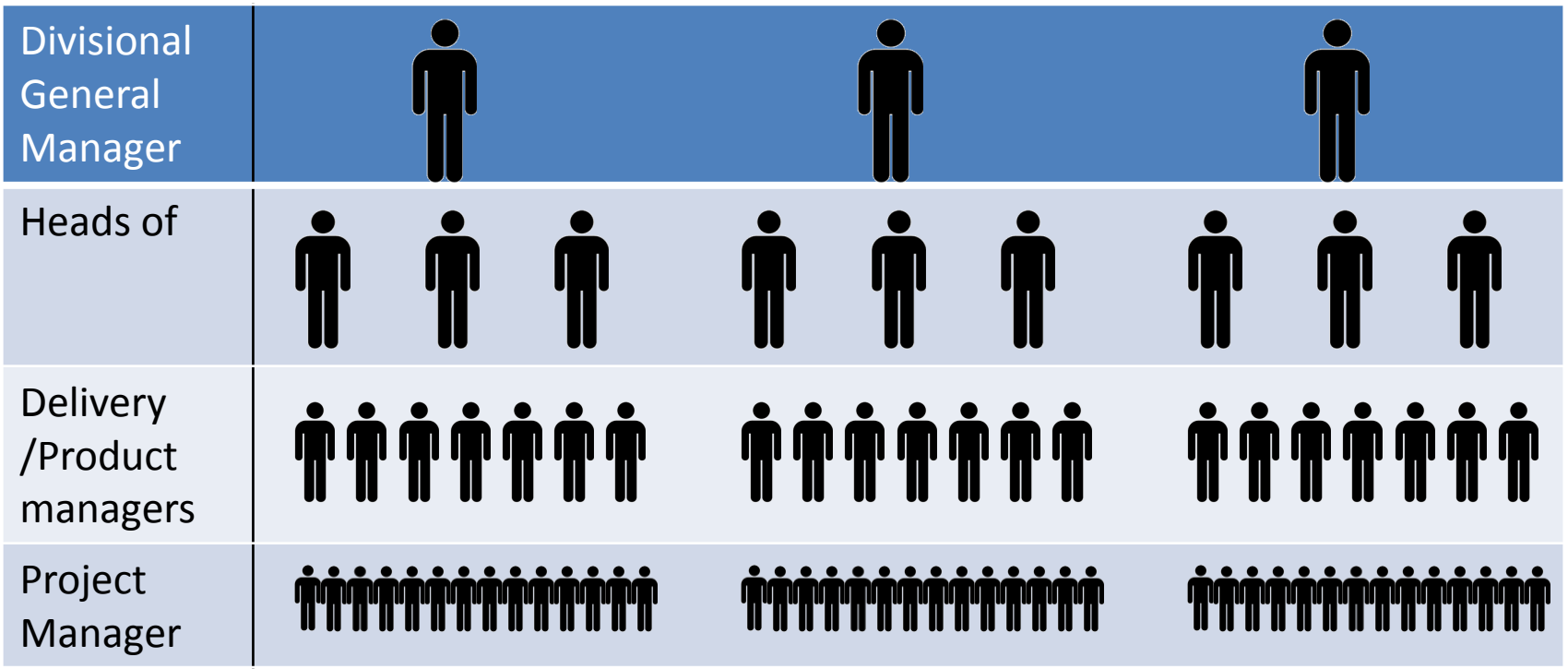


*** Extra workload on top of planned items (a sprint never ends...)**

Operations: One big bottleneck



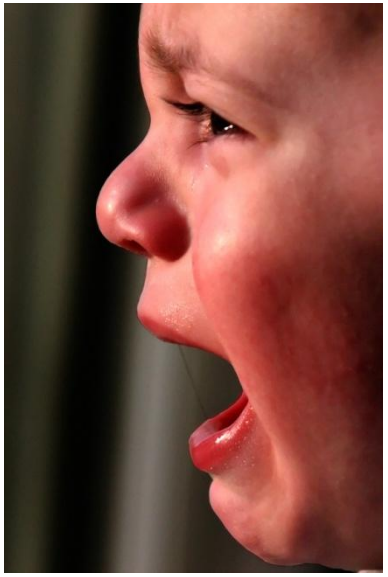
Official Communication



3 main issues for back end specialist team:

- Division heads do not necessarily have the expertise
- Prioritisation via Chinese whispers
- Time delay for decision making

So... if it's urgent?



The crisis-loop

- Desperately holding on to Scrum
  - Stakeholders have lost trust
  - Technical debt increasing
  - Work not done until urgent
  - Silo expertise
  - Management by manouvre

In summary

- Awesome team
- Running hard to stand still

- A ‘victim’ to its environment and corporate structure

APPROACH

How to go about this?

- Others had gone through same thing and left
- Pressure
  - Make change NOW
  - Look like the expert
  - Save the day!
- Highly specialised area: how could I know what was wrong?
  - Decided to observe first

Observation time

- I looked like an idiot

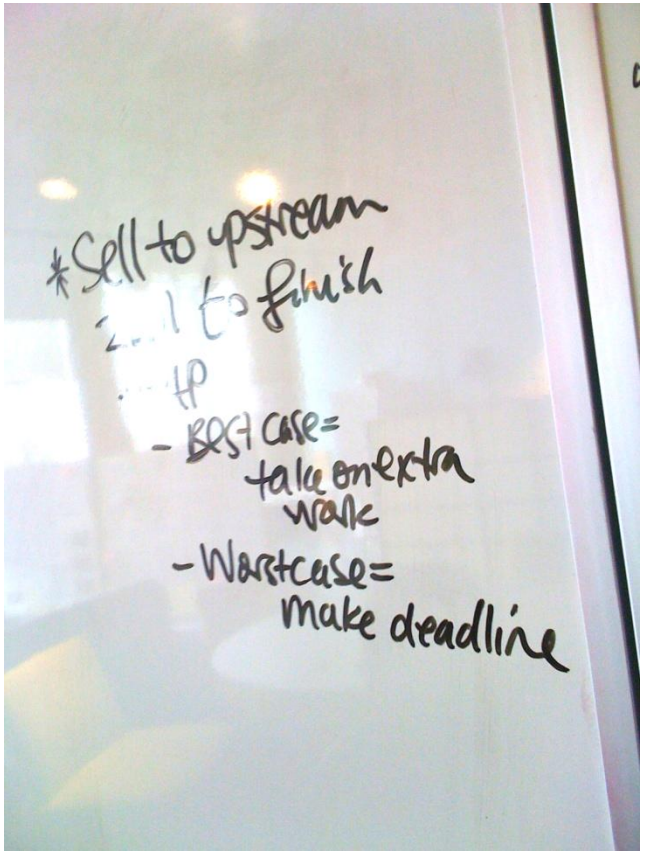
Observations after 3 weeks

- They were making all their commitments last minute BUT
  - “Reliance on 'hero' effort is the norm!
  - Team is EXHAUSTED
  - WHY?????

Causes

- Over 60% of team sprint activity = live and unexpected issues
- Actual time on planned work is at 10% of management expectation
- Struggling with stakeholder liaison - no visibility of progress
- Bugs taking 70 days turnover
- Acceptance Criteria non existent
- Already 6 month plus backlog
- Reviewing 20 more additional requests of work per week
- Capacity falling (ppl leaving)
- Difficulty hiring: specialist knowledge

New culture:
Under promise / Over deliver



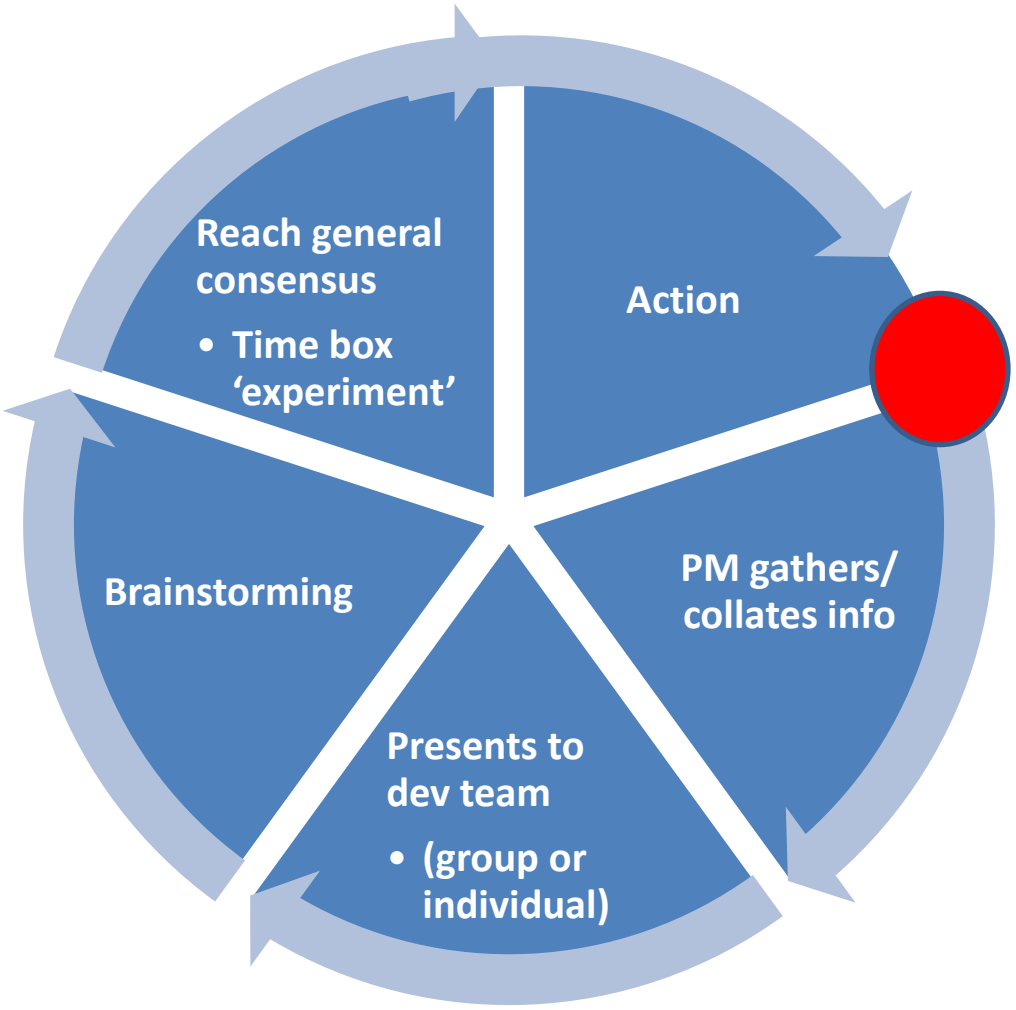
Ask the EXPERTS what to do

Hand the problem over to the REAL problem solvers: those doing the work!

– THE ENGINEERS!!!!!

(Warning to Managers: most engineers are more qualified at solving problems than you are)

Solve problems collaboratively



Change through collaborative experimentation

- Define agreed timeframe
- Action
- Review
- Keep/try something else

THE USUAL EXPLANATION: SCRUM & KANBAN

Kept some Scrum

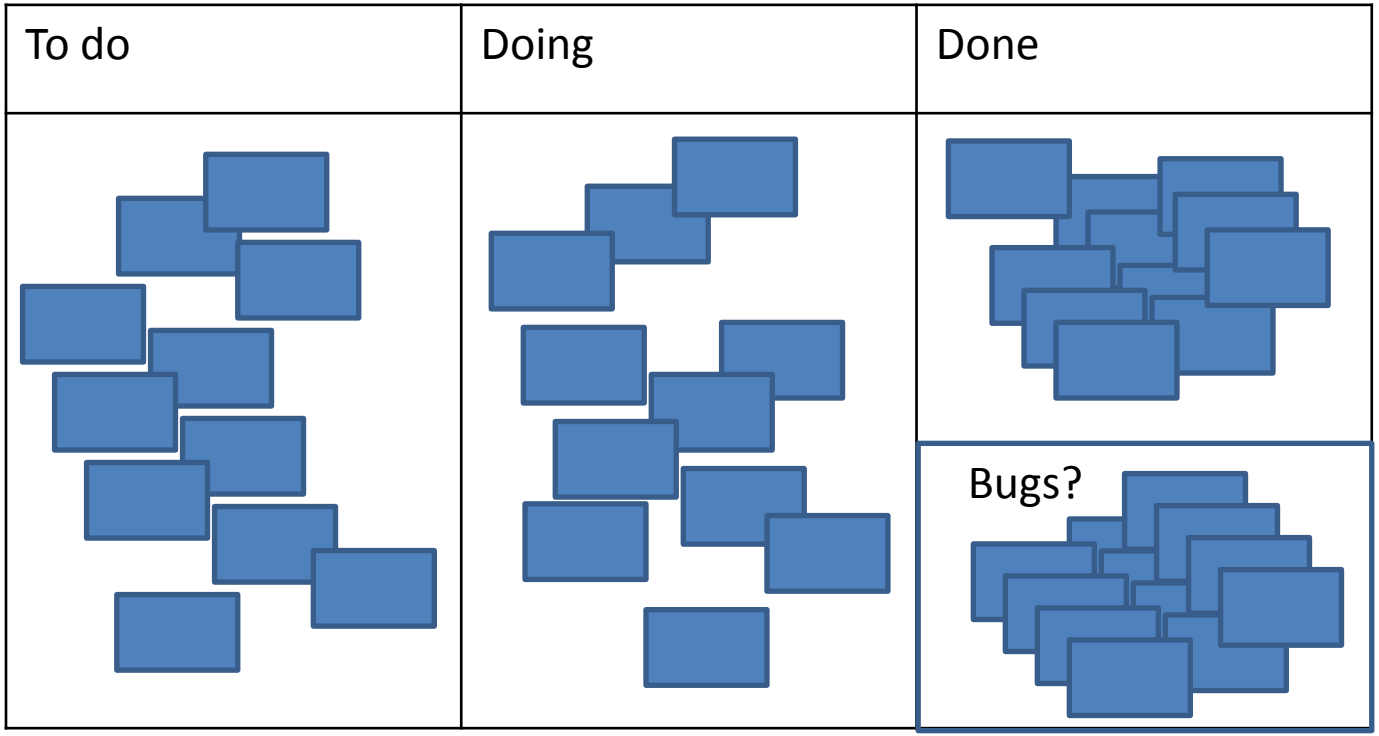
- Kept Scrum just for 40% workload (planned delivery)
  - Matching the rest of the org
- Kept meeting templates
  - But didn’t always use them ‘in the right way’

Did ‘minimal’ Kanban

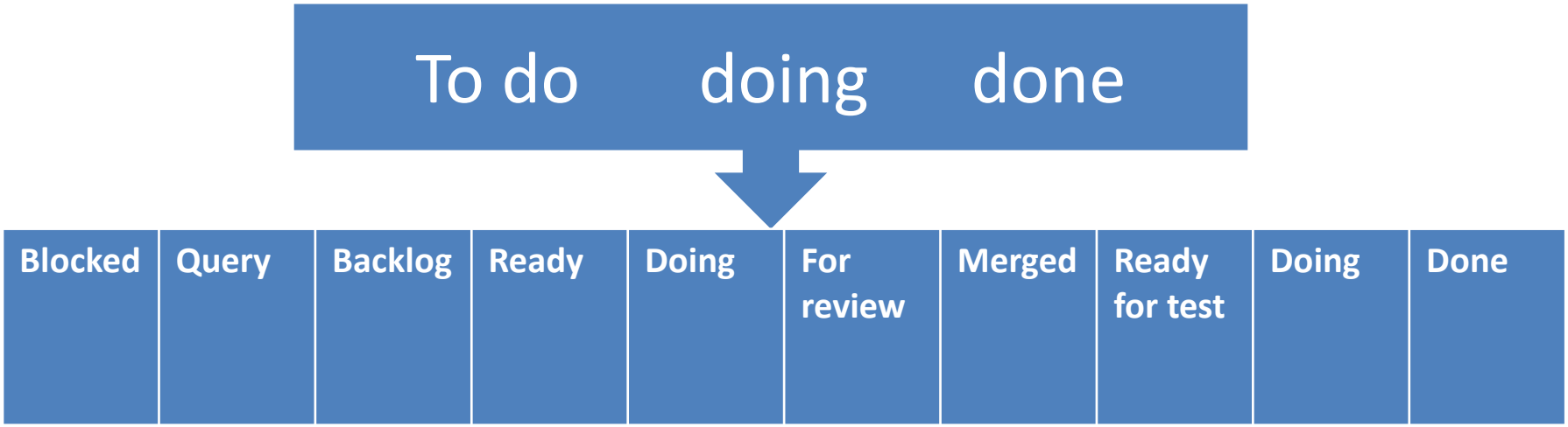
- Observed
- Visualised
- Incremental improvement after observations of patterns

- No ‘proper’ measures
- No fancy graphs or charts

The original ‘day board’



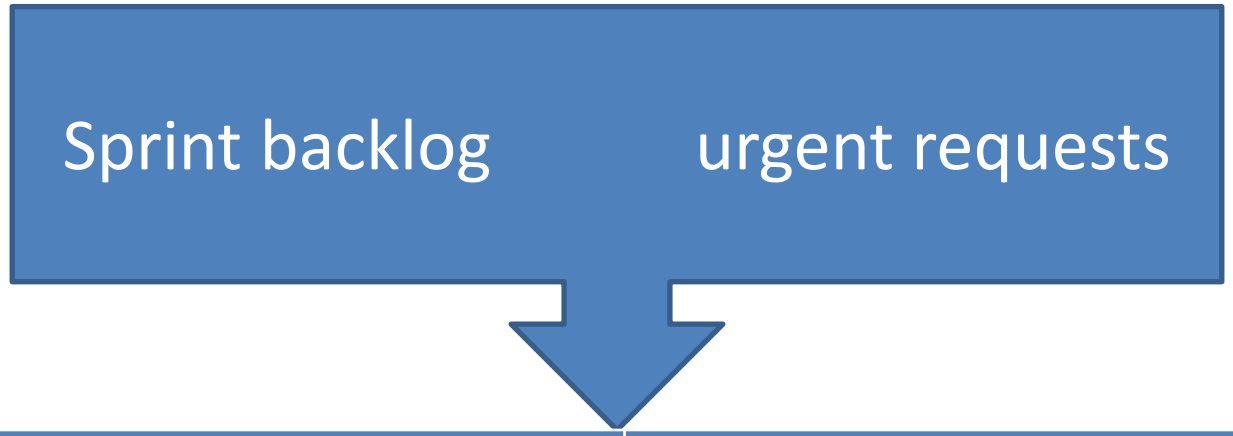
Most requested:
What state is the work actually in?



Onto the day board....

Blocked	Query	Backlog	Ready	Doing	For review	Merged	Ready for test	Doing	Done

What are we working on?



Type	Response needed
Bugs	Days
Planned work	Every two weeks ideally against a 6 month plan
Performance & Optimisation	Indefinite
Technical Debt	Indefinite
Operations development	Kneejerk (hourly?)

Ring fenced reality

Type of work

Actual capacity

60%

Response team

Bugs (1-3 days)

Ops-Dev (now)

Performance & Optimisation

40%

Delivery team

Slate: Planned Delivery

Slate: New solution?

- And then, incrementally improved

- 40% = Delivery team = Scrum-style
- 60% = Response team = Kanban style

USUALLY PRESENTATION ENDS HERE....

In 3 months

- Results
  - Live issues down (60% to 10-20%)
  - Met delivery schedule thus far
  - Most viewed program on iPlayer = no blip
  - Improved stakeholder liaison
    - From Red to Amber for Test and iPlayer (day to day operations, not slate)
    - Online and physical visibility of progress
    - Bugs from 70 days to less than a sprint turnover

AND THAT'S IT????

- REALLY???
- Was that all it took?
- A bit of methodology?

HELL NO! Don’t be fooled

- Its not about the methods, its about people
  - (and if you don’t believe me, read everything from Alistair Cockburn, twice)

- For example
  - Boards/Visualisations etc represent human interactions

  - Meetings / gatherings in Scrum are people collaboration ‘tools’

WHAT WE DID ‘BEHIND THE SCENES’

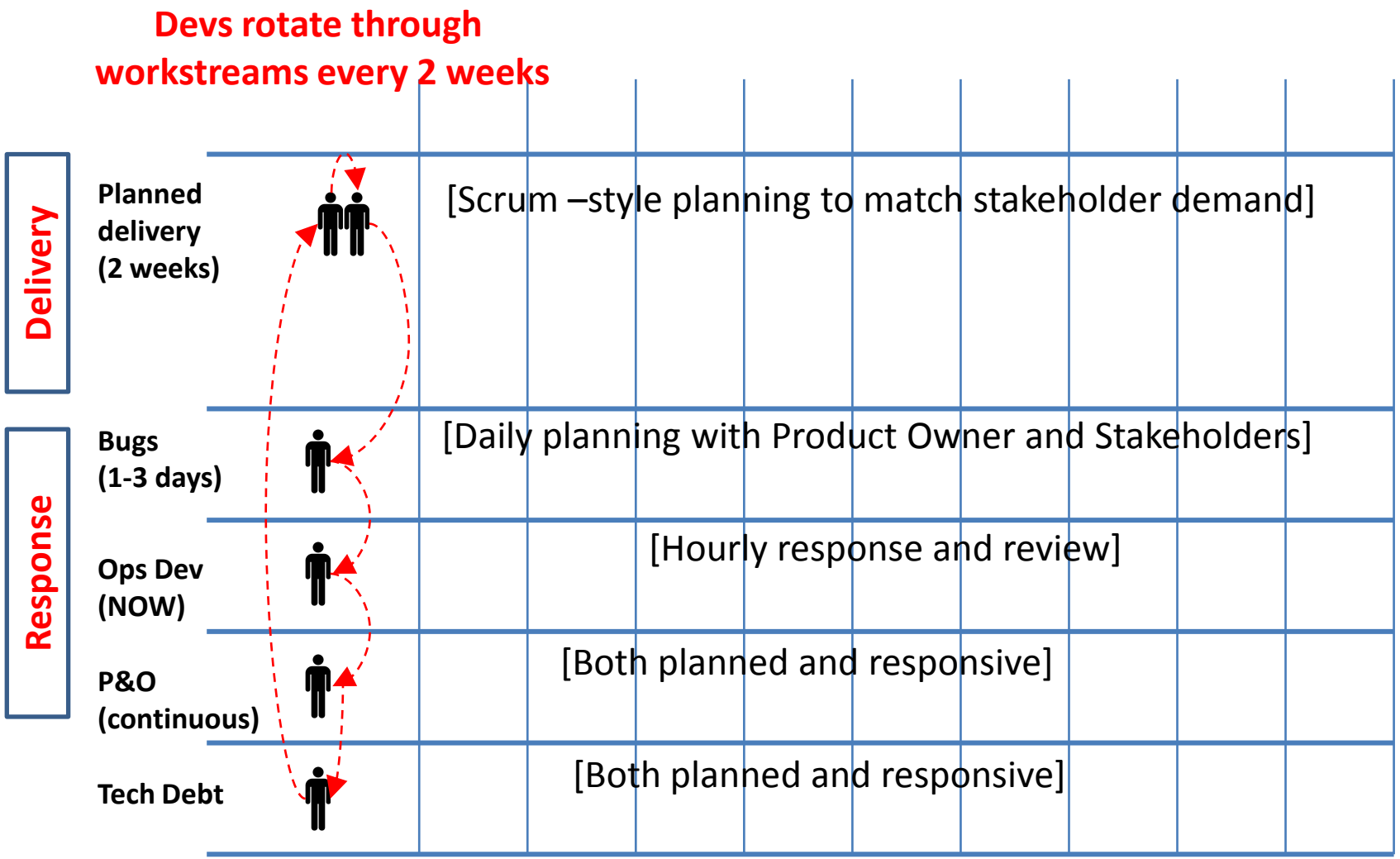
Collaboration

- We concentrated very hard on working together openly and truthfully
- It was HARD work
- It was counter intuitive
- It didn’t feel comfortable
- Some people really struggled with it at the start

Examples

- Quirky stuff we did together
  - Resulted from collaborating
  - Rather than following methodology instructions

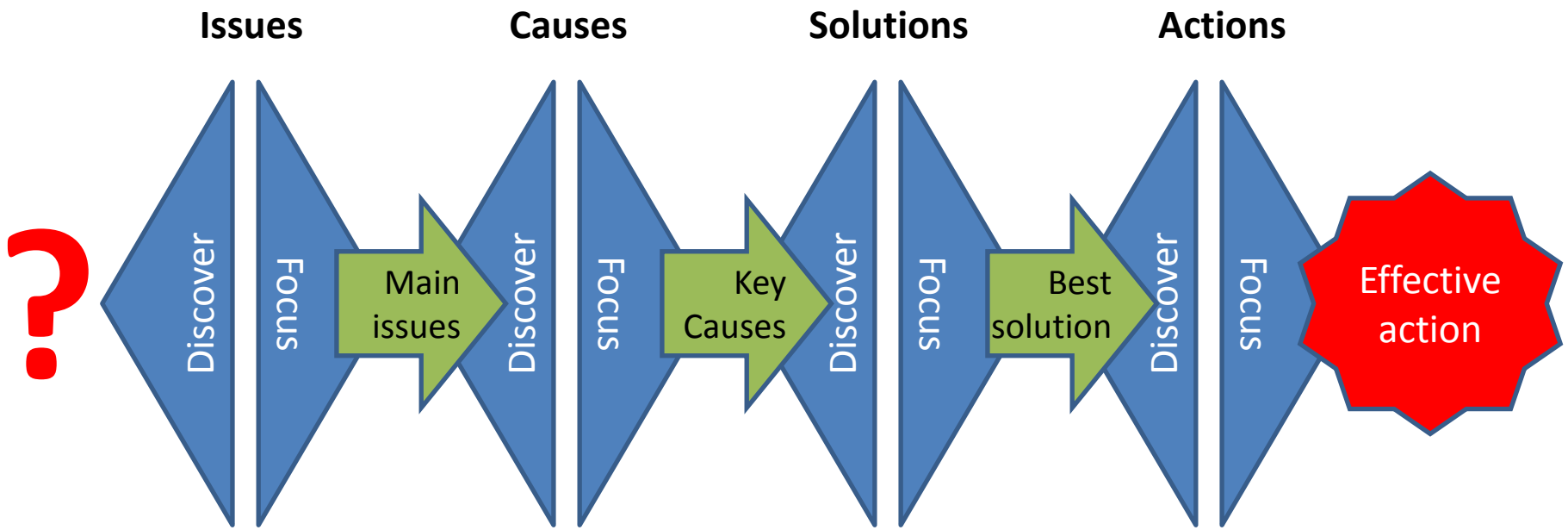
Workstream Methodology Mix-n-Match



Benefits

- Fairness
- Removing ‘single points of failure’
- Distributing knowledge throughout the team
  - Holidays
  - Sickness
  - Mentoring
- Understanding of impact of coding practices

Changed the way we communicated: Expand/Contract*



- Expand: what's wrong?
- Contract: what's the main issues?

- Expand: what's possible causes?
- Contract: what's the main causes?

- Expand: how could we fix this?
- Contract: what would make the most effect?

- Expand: how should we go about this?
- Contract: what, timeframe, how, who?

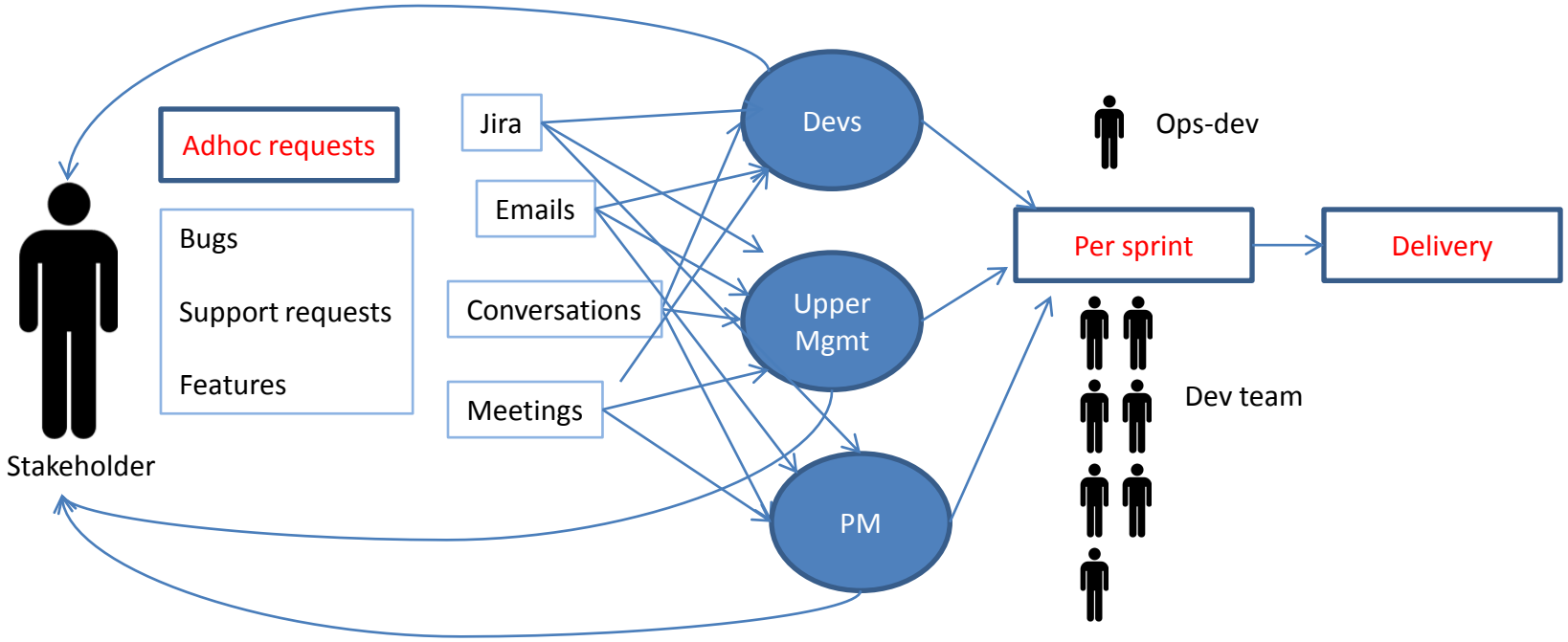
*Rachel Davies knows a lot about this

In everything we did

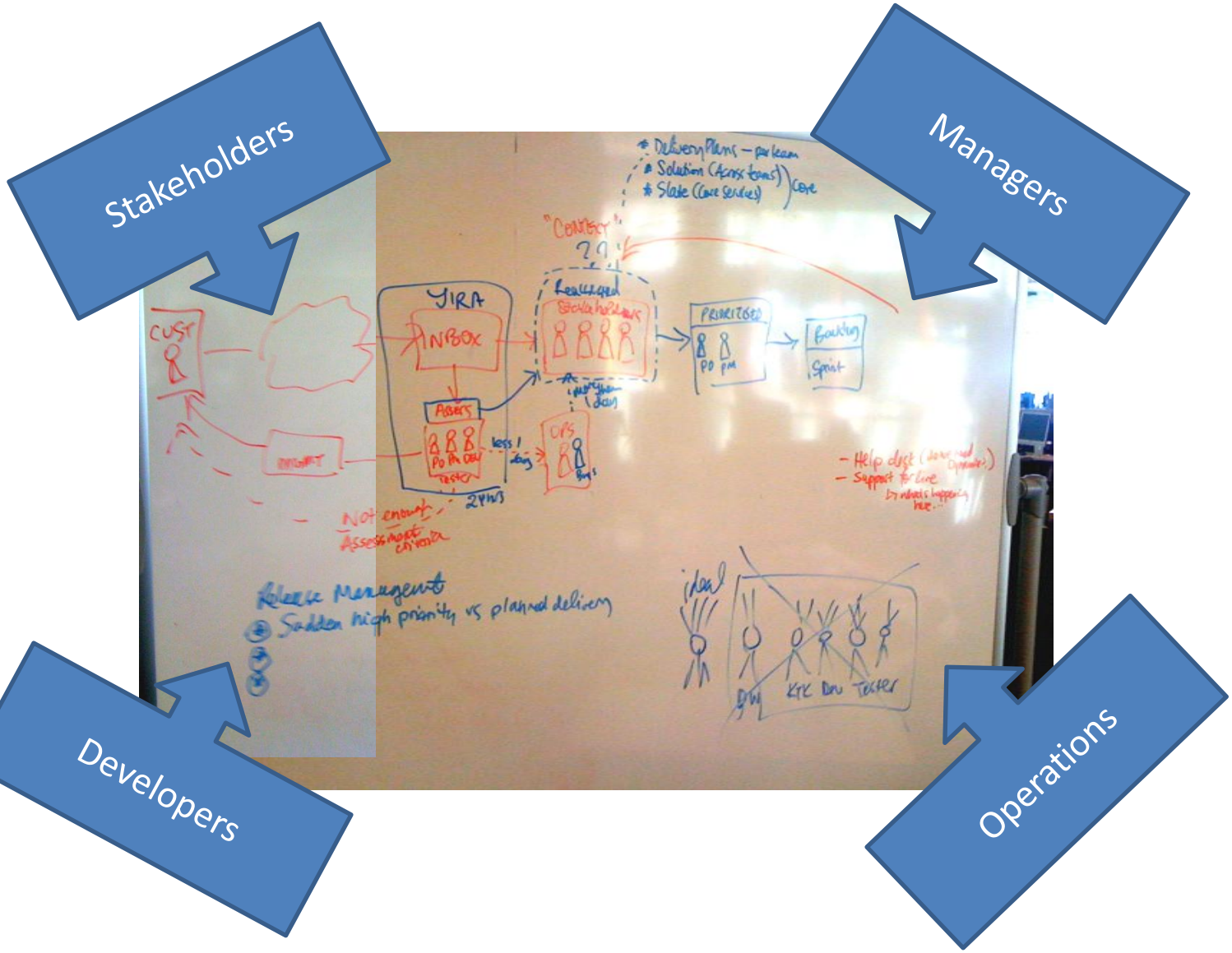
- Conversations
- Reviews
- Retrospectives
- Speculations

Issues – Causes – Solutions - Actions

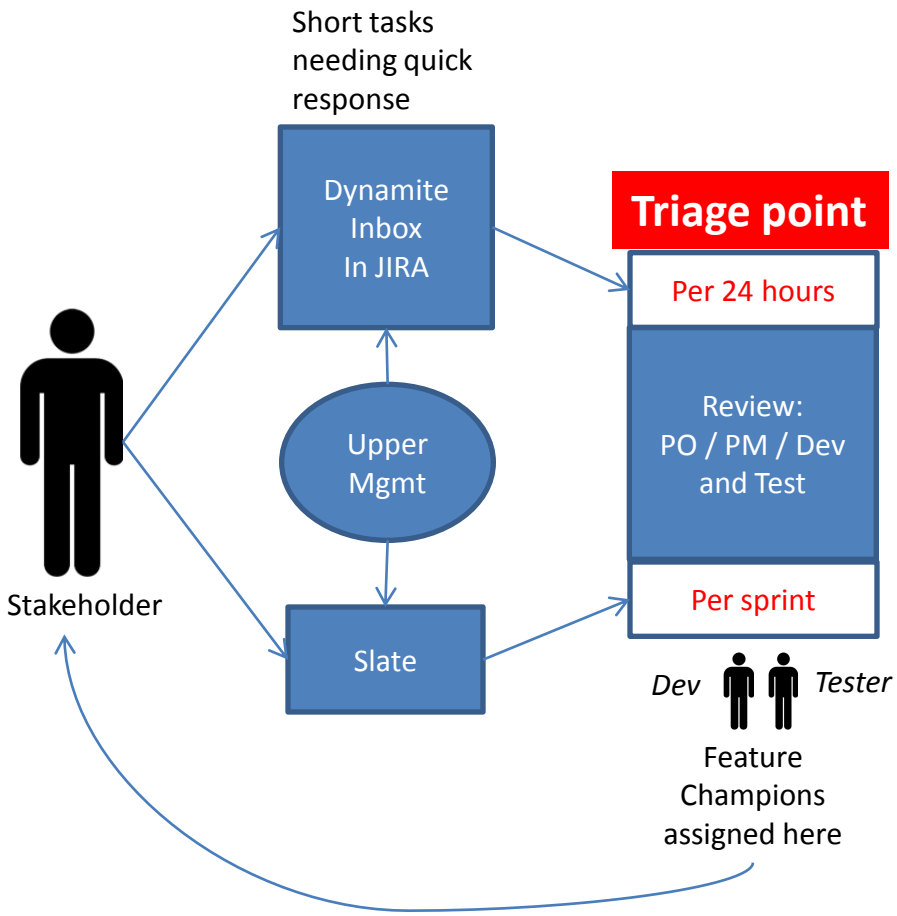
Examine the ‘truth’ openly



Collaborative discussions result

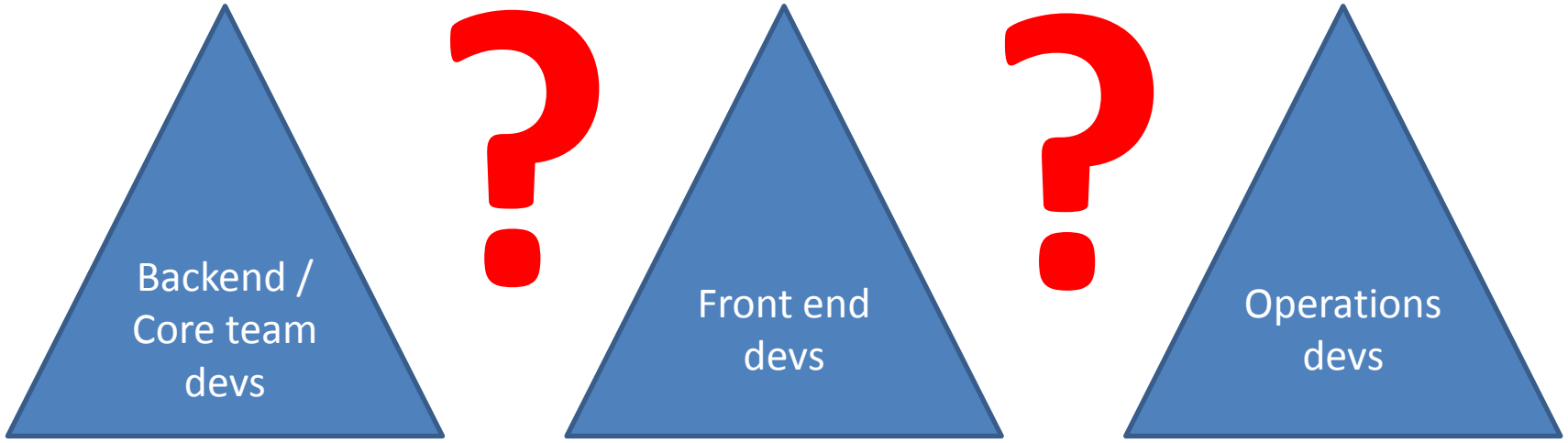


Stakeholder liaison: new set up



BONUS – solving issue by collaborating means we already have buyin

Overcame: Expertise silos



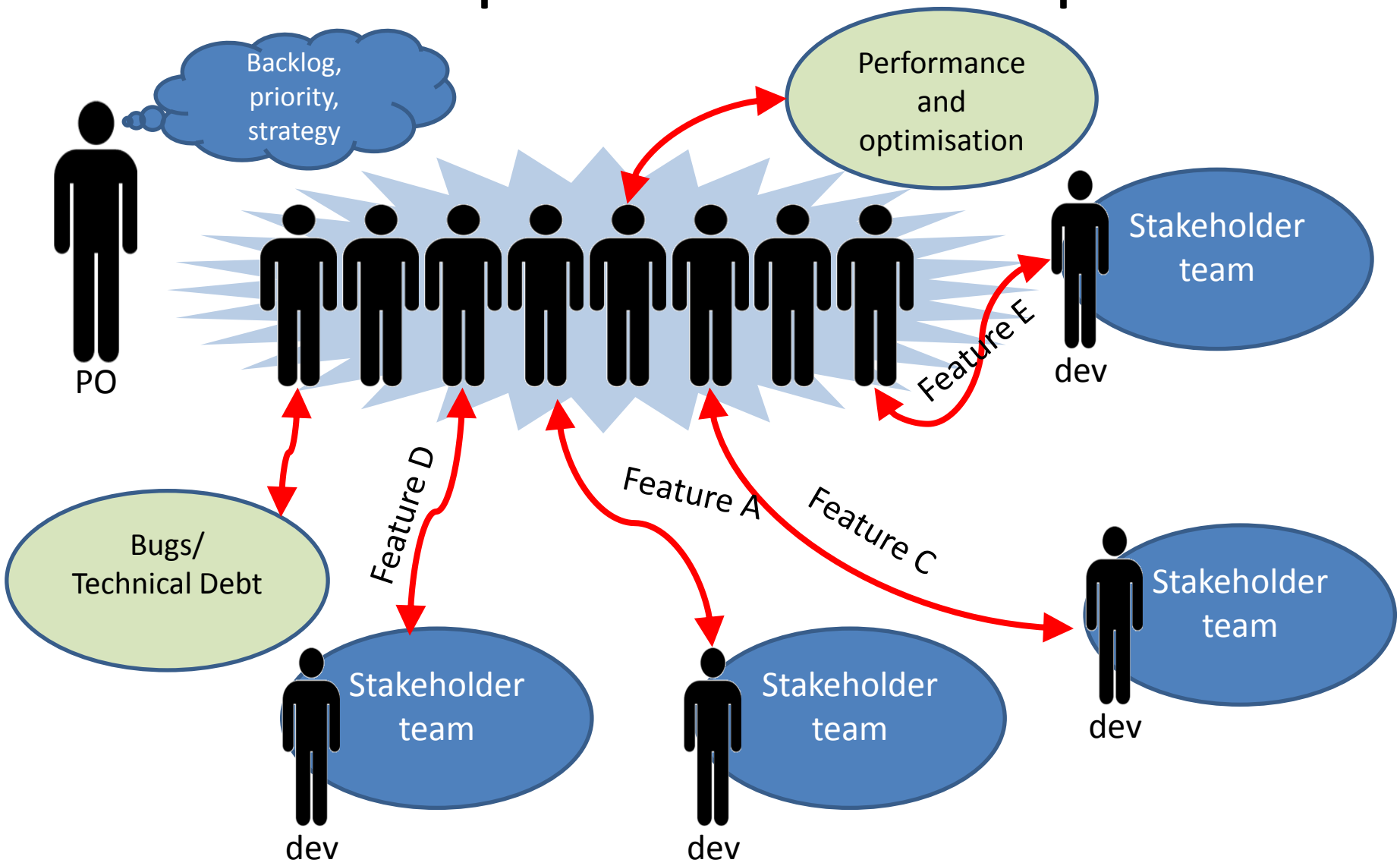
3 main issues for all:

- Integration
- Writing requirements/requests
- Understanding (what each other has done/why)

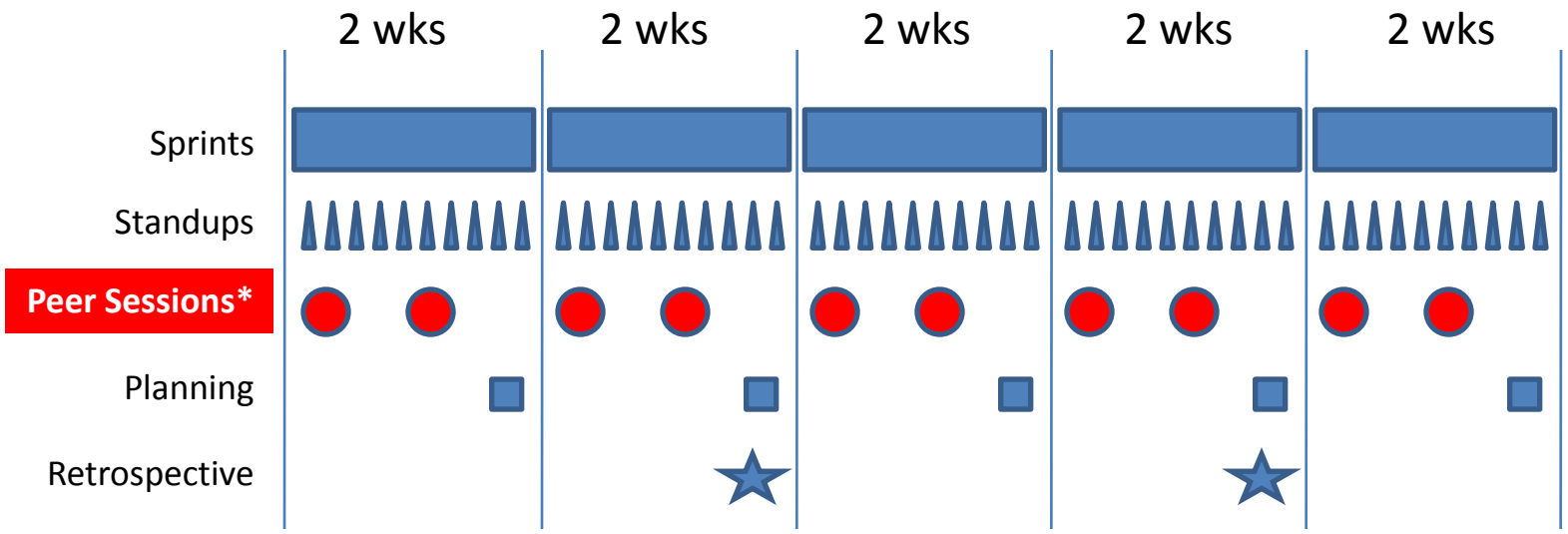
Champions

- Strategic, ‘inner’ PO role
  - NOT a ‘dogs-body’
  - Keeps the overview
  - Responsible for a feature or area of the app
    - Inception > live > maintenance and documentation
  - QUALITY: What / how / when to code
  - Direct liaise with stakeholder devs
  - Breaks down work for backlog if required with PO
  - Reports on progress
  - Involved spearheading realistic estimation

Champions:
REAL product ownership



Initiated Team Peer Sessions



* Optional: Estimating / review / info sharing

Standups – Kanban style
- Issues only
- Info sessions after, if required
- Blocked / hold resolution ASAP
- right to left

Peer Sessions (optional)
- Information transfer
- Feature champion led
- All on same page
- Data to the team (engagement)
- Strategy / plan comms
- Estimation of large features
- Reviewing effectiveness/ capacity

Planning
- Assign support team
- Rotate duties
- Estimation of support work
- Review/resolve operations issues

Defined ideal in REAL words

Ideal	Example of measure of ideal
Increased quality	• no hemorrhaging bugs, last minute surprises and live issues; significant reduction of usage of dev for the 'bugs' role per sprint
Significant reduction of technical debt and it's effects	• Time for refactoring is valued and provided • Refactoring has clearly been done • No 'cowboy' workaround pressure from Product Managers or upper management
Significant reduction to backlog of planned work	• work only on what is required • Jira backlog only contains relevant and organized tickets
Good tracking of current and upcoming workload	• no sudden surprises – e.g. B2B
Increased adaptability	• we can bend and flex with demand: technical solution, devs, testers and process
Increased predictability	• on time delivery for committed items
Commitment process is realistic	• no promising by upstream of what we are not likely to deliver on time – consultation with team/PMs BEFORE commitment
Realistic input and direction from upstream management	• discussing not just what to do, but also HOW – incorporating capacity limitations
Trusted PM/Dev/Tester/upper management relationship	• request from upper management or stakeholder is translated effectively, and efficiently flows through the system with a quality output
More transparent upper management activities	• what's coming up is clear to the team and stakeholders
Happy stakeholders	• effective stakeholder expectation management: bravery to communicate capacity limitations and other commitments • good communication of process, progress on items and outward documentation - example: business friendly release notes
Engaged and empowered devs	• all devs currently in position are retained, and scores of 'job satisfaction' is around 7-8 out of 10, with 85% of all devs indicating

- Simple solutions
- Effective – for our context
- Not in the rulebook
- But in line with the principles of Agile/Lean

REAL RESULT

As we said before: In 3 months

- Results
  - Live issues down (60% to 10-20%)
  - Met delivery schedule thus far
  - Most viewed program on iPlayer = no blip
  - Improved stakeholder liaison
    - From Red to Amber for Test and iPlayer (day to day operations, not slate)
    - Online and physical visibility of progress
    - Bugs from 70 days to less than a sprint turnover

BUT: for the next 3 months

- WITHOUT a manager or coach
- Team self managed
  - Kept improving
  - Didn’t fall back into crisis
  - Kept good stakeholder relationships

18 months later

- From all reports, the team is still going strong
  - Now have a project manager
  - Haven’t fallen back into crisis

Empowerment

People solving problems together

=

Learning

=

Can solve problems on their own

=

Less handholding/time wasting/cost!

REFLECTION

Summary

- Although we did
  - Scrum-ish
  - Kanban-ish
- Why did it work?
- Here is a hint....
  - Individuals and interactions (over processes and tools)
  - Customer collaboration (over customer negotiation)
  - Responding to change (over following a plan)
  - Etc..

Agile/Lean is not a method

- Kanban and Scrum are Agile/Lean
  - **But** Agile/Lean are not necessarily Kanban or Scrum

- The principles **can save 'difficult' projects**
  - **Even when methods can't**

- Use principles as your guide
- Reality as your driver
- And methods as your tools

In a crisis loop

- Suggestion
  - If you have to choose between a process (e.g. Scrum or Kanban) and adhering to Agile/Lean Principles....
  - Choose the principles!

(err... that'd be this one: individuals and interactions over processes and tools)

;-)

RAF GEMMAIL

A Dev's Eye View

We practiced Scrum:

- Sprints
- Pointing
- Planning poker
- XP

But during the Sprint:

- URGENT issues
- Out of remit features

But during the Sprint:

- URGENT issues
- Out of remit features
- Failure to learn from history

Planned work compromised by unplanned work

The climate

- **Code decay**

The climate

- Code decay
- **Reviews blocking features**

The climate

- Code decay
- Reviews blocking features
- **Devs and PM's leaving**

The climate

- Code decay
- Reviews blocking features
- Devs and PM's leaving
- **No time to improve dev process**

The climate

- Code decay
- Reviews blocking features
- Devs and PM's leaving
- No time to improve dev process

09:30 Almost done

10:00 Stand up *"I just have to merge it."*

Merge

Test

11:00 Done

09:30 Nearly done

10:00 Stand up

Merge

Test Failed

Code

Test

Push

11:30 Done

09:30 Nearly done

10:00 Stand up

Merge

Test Failed

Bug: *“Urgent! Who is available?”*

Code

Test

Push

14:00 Done

09:30 Nearly done

10:00 Stand up

Merge

Test Failed

Bug: “Stake holder complained..”

Code

Production Issue

Test

Push

18:00 Done

- *90 mins work*

== 8h day

Katherine Kirk on the Bridge

- You guys are AMAZING
- But Stakeholders are scared

Katherine Kirk on the Bridge

- You guys are AMAZING
- But Stakeholders are scared
- What do you think we should do?

Katherine Kirk on the Bridge

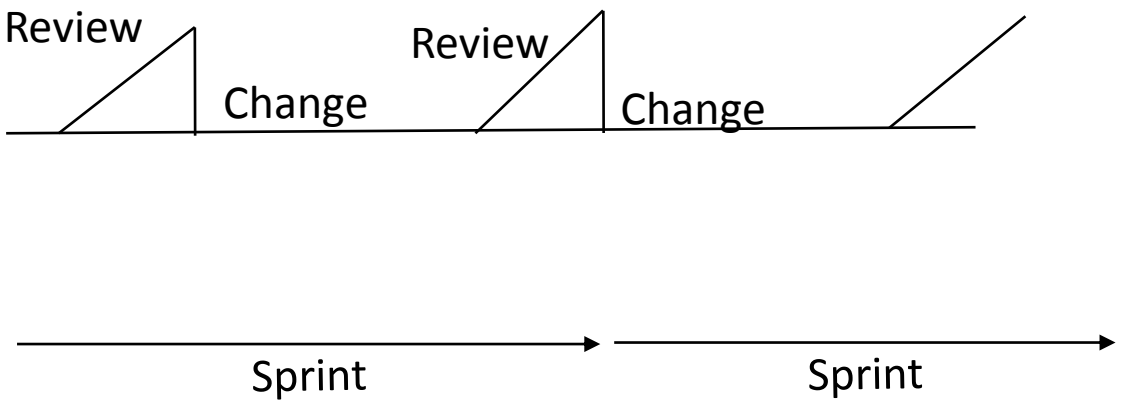
- You guys are AMAZING
- But Stakeholders are scared
- What do you think we should do?

Did she just ask us to fix the PM function??

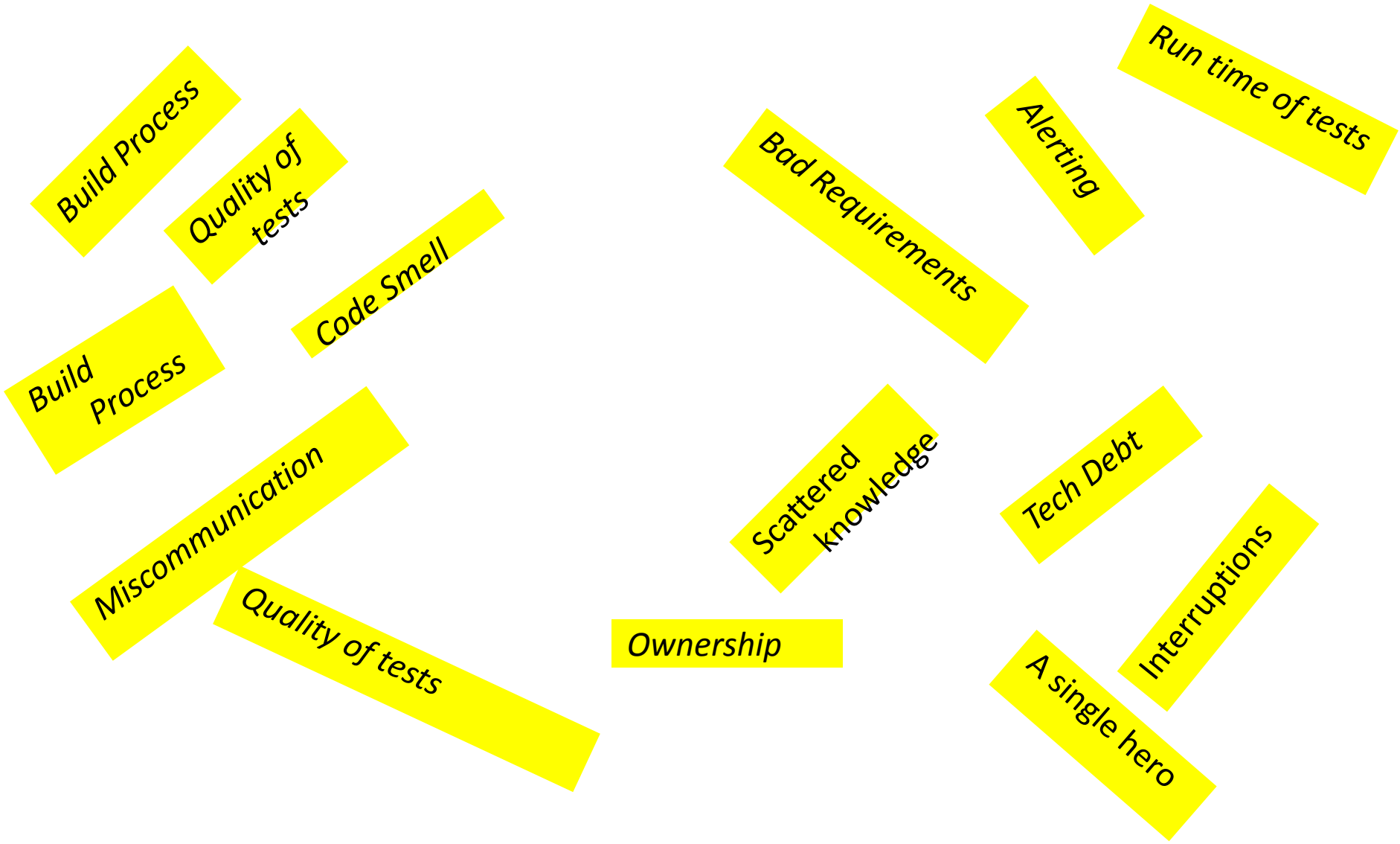
Are the stake holders letting her?

Nemawashi (根回し)

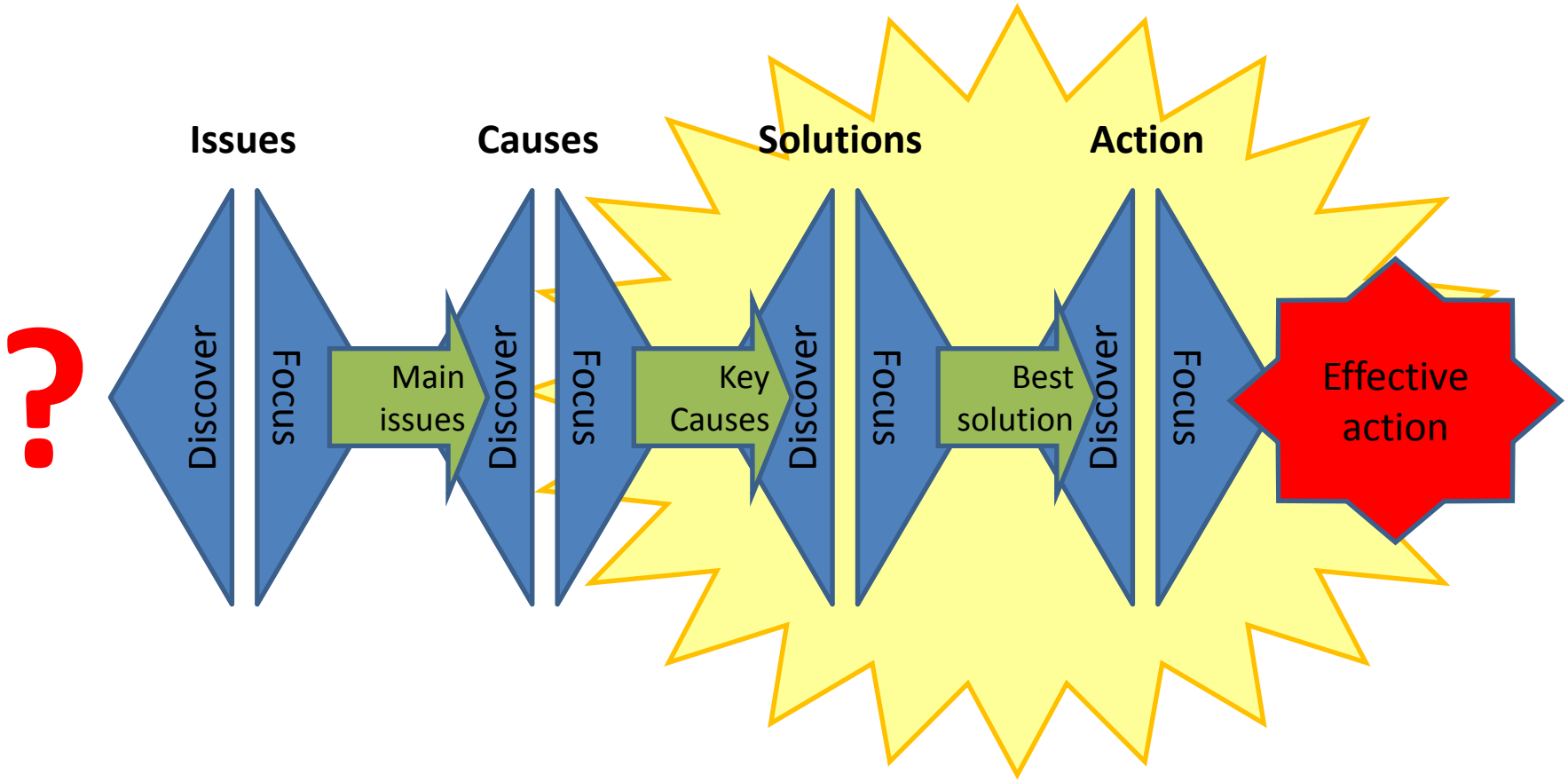
Improve without compromising current workload



The 'normal' Retrospective noise

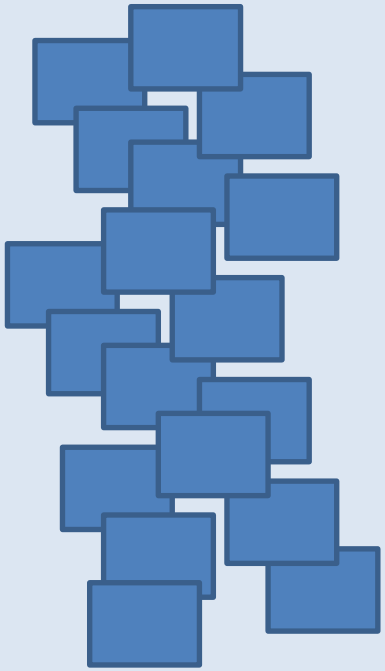


Review: Expand/Contract

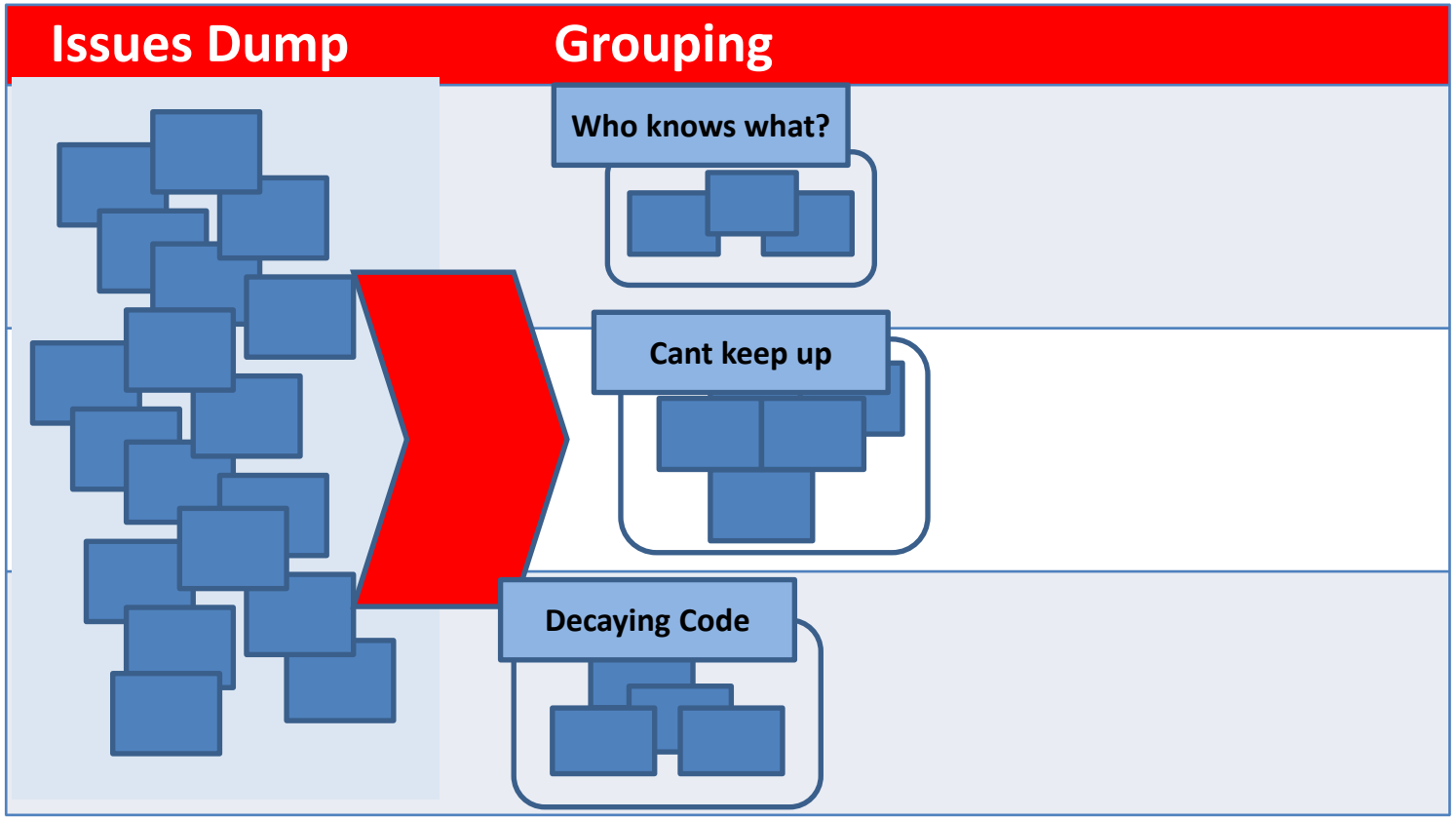


Example

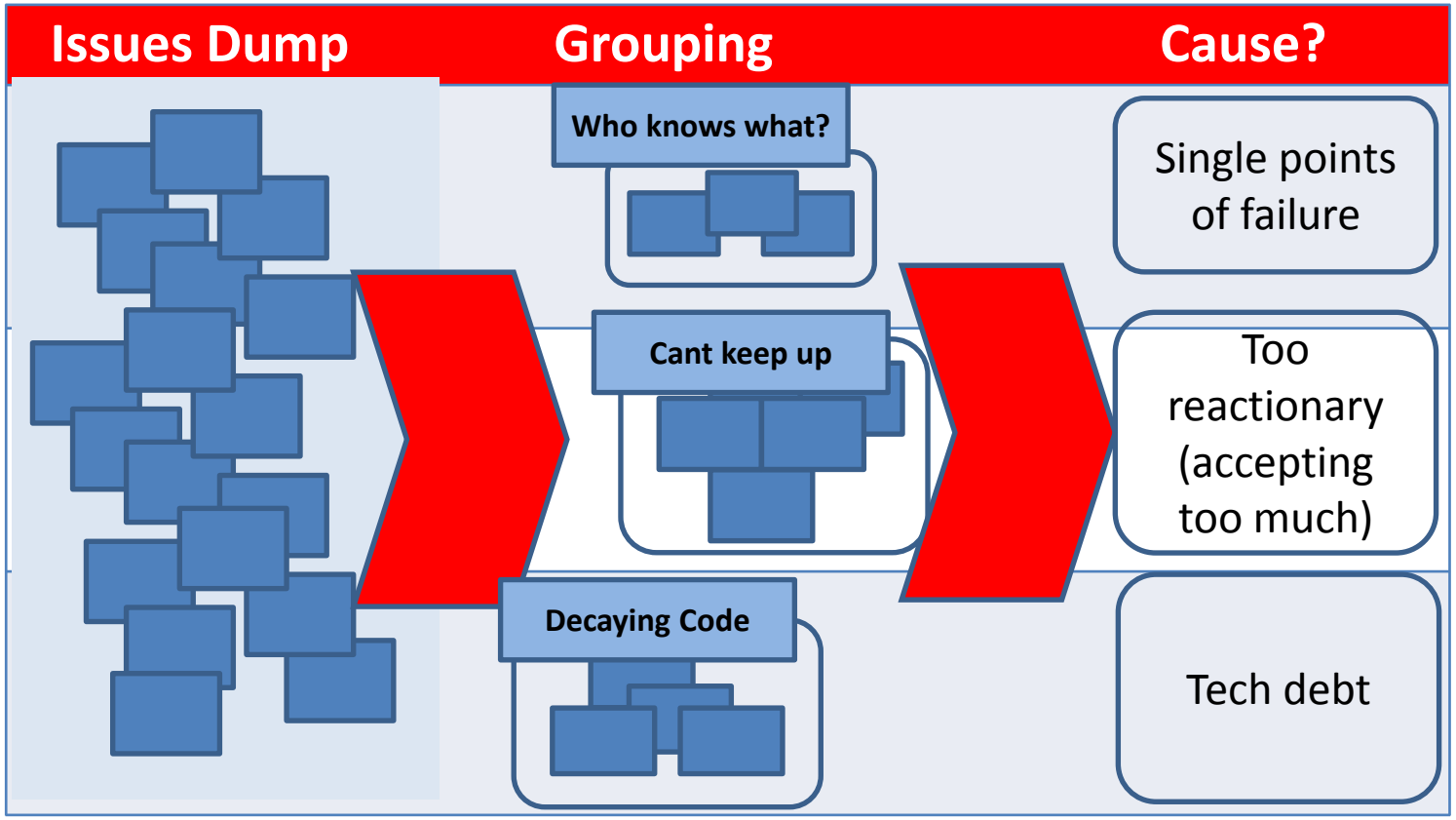
Issues Dump



Example


Issues Dump
Grouping
Who knows what?
Cant keep up
Decaying Code

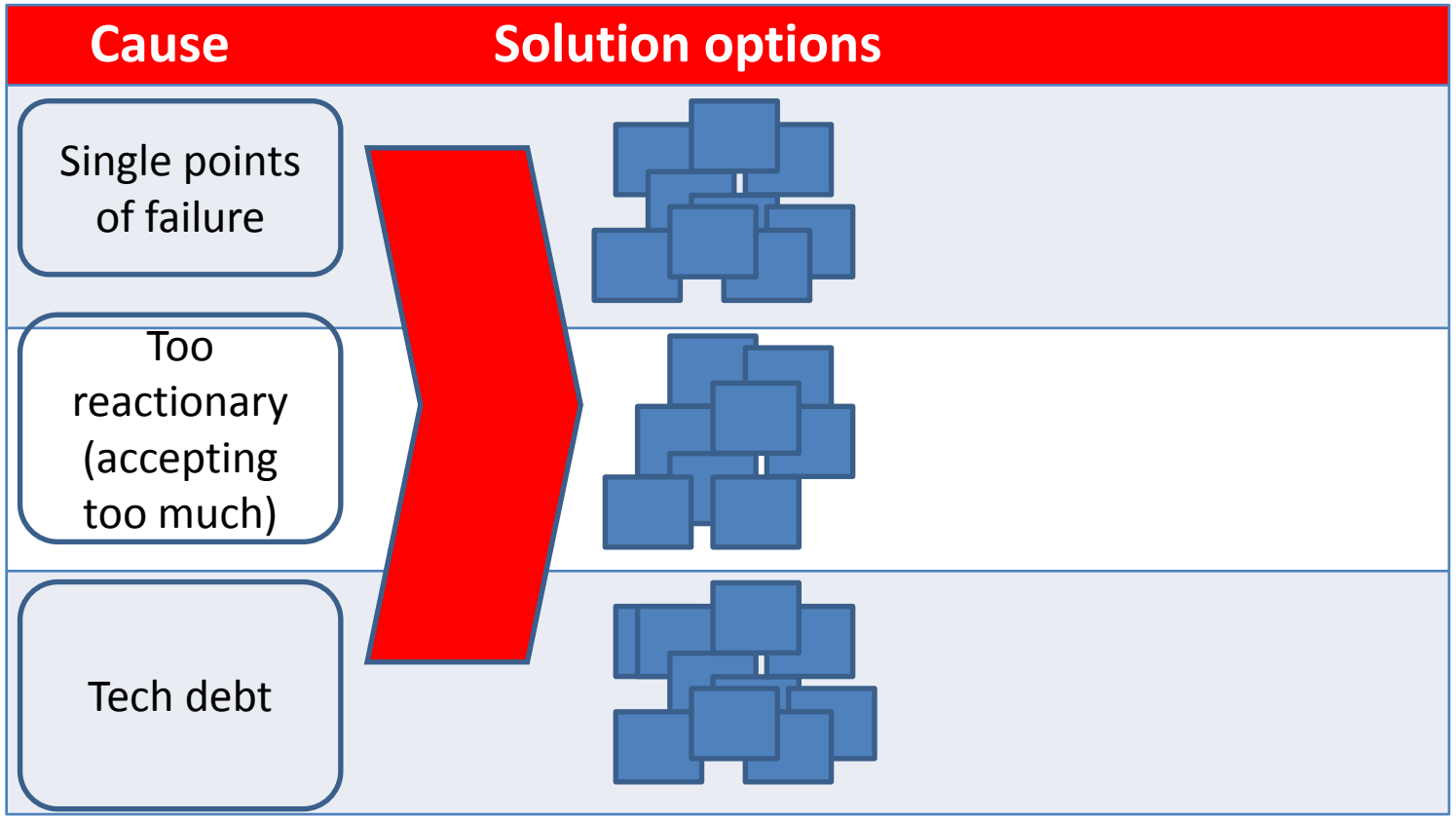
Example


Issues Dump
Grouping
Cause?
Who knows what?
Cant keep up
Decaying Code
Single points of failure
Too reactionary (accepting too much)
Tech debt

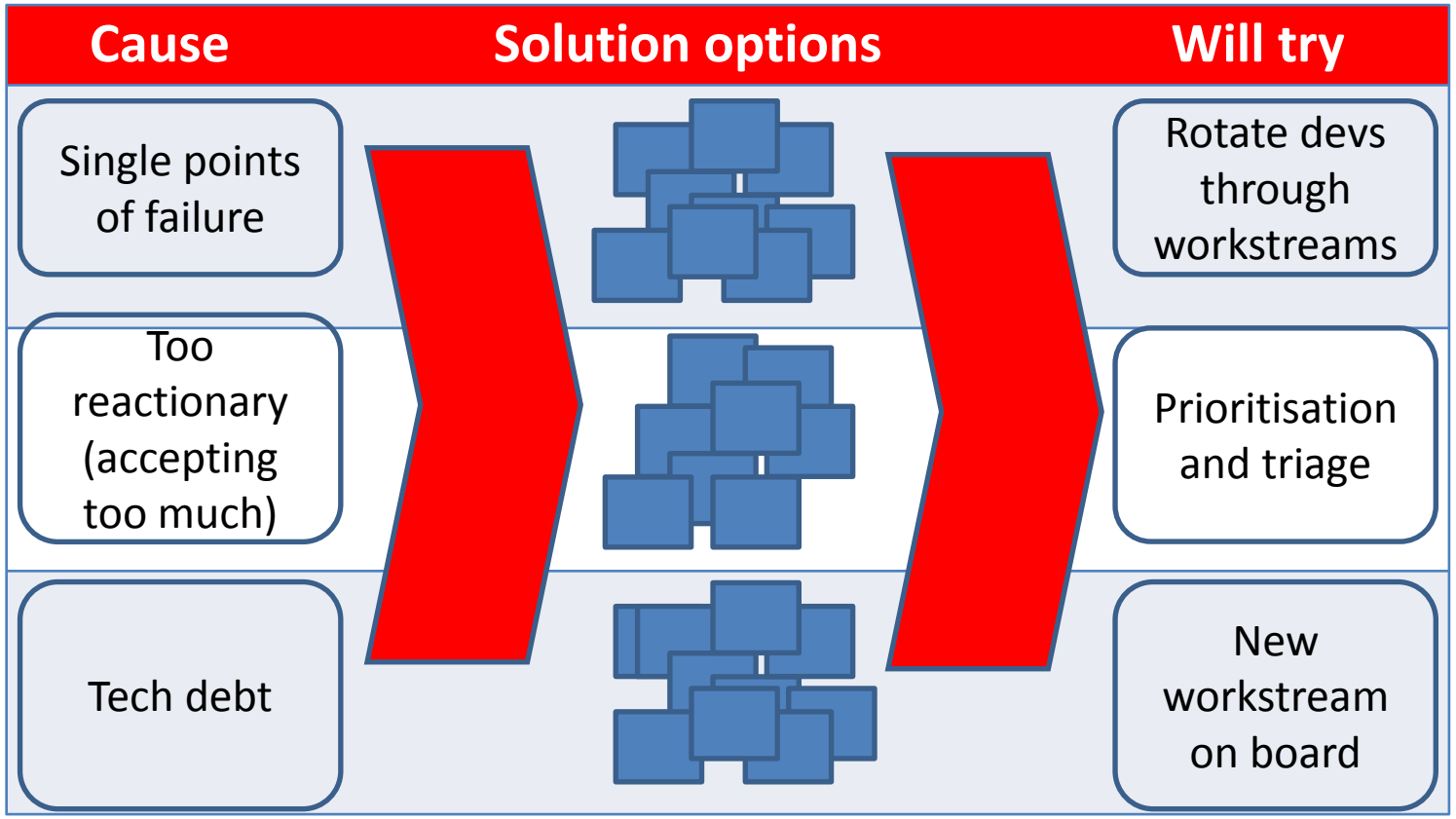
Action

Cause
Single points of failure
Too reactionary (accepting too much)
Tech debt

Action


Cause
Solution options
Single points of failure
Too reactionary (accepting too much)
Tech debt

Action


Cause
Solution options
Will try
Single points of failure
Too reactionary (accepting too much)
Tech debt
Rotate devs through workstreams
Prioritisation and triage
New workstream on board

No more heros

- A reactive Pull-based Response Team
- Feature Champions to PO critical features
- An Empowered Team!

Response team:

- Bugs
- Ops
- Performance and optimisation

'Everyone-is-a-Hero' Rotation

Ops

Bugs

'Everyone-is-a-Hero' Rotation

Ops

- Release Process
- Technical Debt
- Process automation
- Stability

Bugs

'Everyone-is-a-Hero' Rotation

Ops

- Release Process
- Technical Debt
- Process automation
- Stability

Bugs

Burdensome

Needs to be done

Often user error

'Everyone-is-a-Hero' Rotation

Ops

- Release Process
- Technical Debt
- Process automation
- Stability

Bugs

Burdensome

Needs to be done

Often user error

Shared Knowledge

Planned work: A new day!

1 days work
== 1 day uninterrupted work!!!!!

- 9am: Work on feature – include some TD
- Stand up
- CODE (Review / Have code review)
- TEST (Test – Merge – Test – Push)
- 1800: HOME

Response work: A new way!

1 days work == whatever needed!!!!!

- 9:30am Check Splunk Alerts
- 10am Stand up
- 10:15am Pull P&O card
- 12pm Discuss optimisation with Recommendations team
- 2pm Pair with TL on incident
- 3pm Review Related Code and raise ticket
- 4pm Refactor and speed up some feed

Visualisations provided a more granular understanding

- **Dev = {analysis, dev, review, testing, merge}**

Visualisations provided a more granular understanding

- Dev = {analysis, dev, review, testing, merge}
- “***I'm nearly done***” → “***He's in review***”

Visualisations provided a more granular understanding

- Dev = {analysis, dev, review, testing, merge}
- “I'm nearly done” → “He's in review”
- “***I'm merging***” → “***Dev's still got tests to run***”

Visualisations provided a more granular understanding

- Dev = {analysis, dev, review, testing, merge}
- “I'm nearly done” → “He's in review”
- “I'm merging” → “The dev's still got tests to run”
- **Test Column → Test Board**

Visualisations provided a more granular understanding

- Dev = {analysis, dev, review, testing, merge}
- “I'm nearly done” → “He's in review”
- “I'm merging” → “The dev's still got tests to run”
- Test Column → Test Board

Self Management

- Continued to improve “established” process
- Experiments with pointing
- Moves towards pure TDD
- New PM → went to Scrumban

Communication & Collaboration
Over
Process

On Reflection

Consider

- If we'd tried
  - Scrum-right
  - Kanban-right

- Not so Agile/Lean?
- Results as quick?
- As Sustainable?
- Self-managing?

Principles

Lean

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

Agile Manifesto

Individuals and interactions over processes and tools

Working software over comprehensive documentation

Customer collaboration over contract negotiation

Responding to change over following a plan

Nemawashi (根回し)