



Ritchie BI

WHAT F1 TAUGHT ME ABOUT DELIVERY SPEED: HOW DATA TEAMS CAN MOVE AT LIGHTSPEED

Big Data LDN 2026

Keynote Summary

Edward Ritchie · Director, Ritchie BI · 24th September 2026

WORK SPENDS MOST OF ITS LIFE WAITING

Data project often fall behind schedule. At the same time, they often spend most of their life idle. On average, projects only move forward for 5%-15% of their duration.

The other 85%+ is time spent waiting between tasks:

- Waiting for a meeting
- Waiting for feedback from someone who is on holiday
- Waiting for someone to validate the results
- ...

It doesn't *feel* like waiting to us, because we naturally find something else we can progress. However, the original work still sits idle.

Teams try to find speed through AI, or by developing technical skills. This predominantly speeds up the 15%, while the 85% goes untouched.

The focus should be less about how we can work faster, and more about how we can wait less. The following slides show how to solve this.



DON'T MOVE IT ALL FORWARD

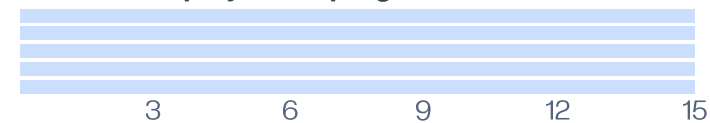
Little's Law: Cycle time = work-in-progress ÷ throughput

If team throughput is 1 project every 3 months, cycle time = $WIP \div 1/3$

If we have 5 projects in progress, cycle time = 15 months:

- No projects are completed before month 15
- Projects are open for 15 months

WIP = 5; five projects in progress



If we have 1 project in progress, cycle time = 3 months:

- First project is completed at month 3
- Every project, except the last, arrives earlier
- Projects are open for 3 months

WIP = 1; one project in progress



What to do

- Finish one thing before starting the next
- Focus on highest priority first, at project level and task level
- Address bottlenecks exposed by low work-in-progress

AVOID BLOCKED CALENDARS

Kingman's formula (right) shows that waiting time increases with utilisation and variation.

As data work is high variation, utilisation needs to be carefully controlled to avoid waiting time exploding. Sir John Kingman, 1962: "In heavy traffic the equilibrium waiting time will be large."

If we plan to 99% of capacity, the utilisation factor in Kingman's formula is 99.0

If we plan to 70% of capacity, the utilisation factor in Kingman's formula is 2.3

What to do

- Don't plan to fill every hour*; keep your calendar like a fast-moving motorway
- Manage incoming requests
- Use 'spare' time for bottlenecks and unpredictable work

*Aim for 60-80% utilisation, towards the lower end if variability is high, and towards the upper end if variability is low.

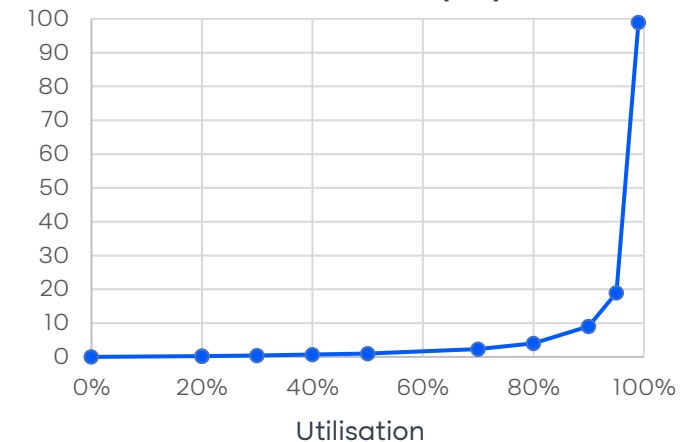
Kingman's Formula

$$E(W) = \left(\frac{p}{1-p} \right) \times \left(\frac{C_a^2 + C_s^2}{2} \right) \times \mu_s$$

$E(W)$ = Expected Waiting Time
 p = Percentage Utilization
 C_a = Coefficient of arrival variation. Standard deviation divided by mean of time between arrivals
 C_s = Coefficient of service (process) variation. Standard deviation divided by mean of service (process) duration
 μ_s = Mean service (process) duration

Expected Waiting Time Utilisation Arrival Time Variation Process Time Variation

Utilisation Factor (p/1-p)



BUILD TEAMS LIKE QRM CELLS

Rajan Suri, the founder of QRM, found that high-mix, low-volume manufacturers were too slow to keep up with customer demand.

The key reasons were functional teams, frequent handoffs, and no end-to-end ownership. Most data teams share the same diagnosis.

The table to the right shows the key differences between traditional manufacturing and a QRM cell. Organisations that adopted QRM reported 75-90% reductions in build duration.

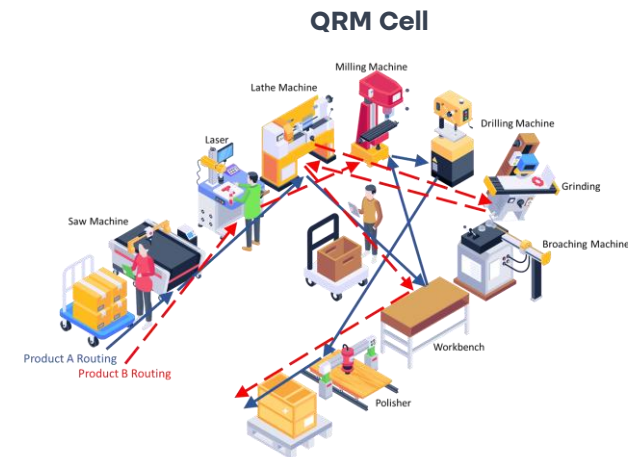
Data teams that apply the same principles will also see significant improvement in delivery speed.

What to do

- Sit close to the business: learn the domain, processes and systems
- Cross-skill: learn whichever handoff is your bottleneck
- Own the outcome: build the puzzle, don't just deliver a piece

Comparison: Traditional vs QRM Cell

	Traditional	QRM Cell
Ownership	Functional role	End-to-end
Team focus	Technical	Target market segment
Handoffs	Frequent	No, cross-skilled



CASE STUDY

After experiencing significant delays on an early project with the F1 team, we applied the principles from ops research to minimize waiting.

1. Don't move it all forward

- Limited work-in-progress, from three parallel projects to one high-priority project
- Sequenced items
- Low work-in-progress exposed bottlenecks

2. Avoid blocked calendars

- Used the 'spare' capacity to address bottlenecks

3. Build teams like QRM cells

- Learned the processes, operations, systems, needs etc.
- Reduced handoffs to bottleneck processes
- Took ownership to mitigate risks

Result

More than 4x faster delivery, normalised for scope. We had the same people and tools; the steps were no quicker, we just waited less.

Examples of time saving

Risk	Action	Est.	Actual
"I'm busy on other projects"	Got familiar with 'new' data platform and learned systems and processes, so could build pipelines with correct logic	3 weeks	2 days
"Not possible to get that data"	Learned the system back-end and found the raw source fields	1 week	1 hour
"That doesn't align with our system"	Learned the system front-end and reconciled data	3 days	2 hours

ONE-PAGE SUMMARY

Work spends most of its life **waiting**

1

"Don't move it all forward"

- Limit work in progress
- Sequence work; finish one before starting the next

2

"Avoid blocked calendars"

- Don't plan to fill every hour
- Manage incoming requests
- Use 'spare' time for bottlenecks and unpredictable work

3

"Build teams like QRM cells"

- Cross-skilled
- Close to the business
- End-to-end ownership