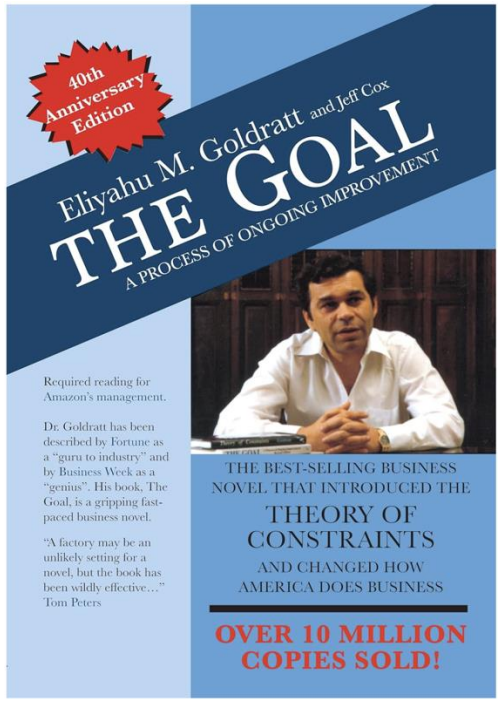
Everyone seems busy…
Why are we still late?

Solving Systems Problems using Systems Thinking:

Finding the Real Constraint

With Simon White – Chair, TOCICO &
Senior Program Manager, WiseTech Global

Theory of Constraints – 30 second History



The Theory of Constraints (TOC) is a management philosophy focused on identifying and managing the single most important limiting factor (constraint) that prevents a system from achieving its goals.

Definition: A constraint is any factor—physical, policy-based, or market-driven—that limits a system's performance.

Key Assumptions

Assumption 1:

All systems are finite, and must therefore have some kind of limit

Assumption 2:

A finite system will have only one constraint, at any given point in time, that is preventing it from doing more

ToC model is based on whats called ‘The 5 Focusing Steps’

The 5 Focusing Steps:

- **Identify** the system's constraint.
- **Exploit** the constraint (make sure it's running at maximum capacity).
- **Subordinate** everything else to the above decision (align non-constraints to support the bottleneck).
- **Elevate** the constraint (increase its capacity).
- **Repeat:** If a constraint is broken, go back to step 1, but do not let inertia cause a system constraint.

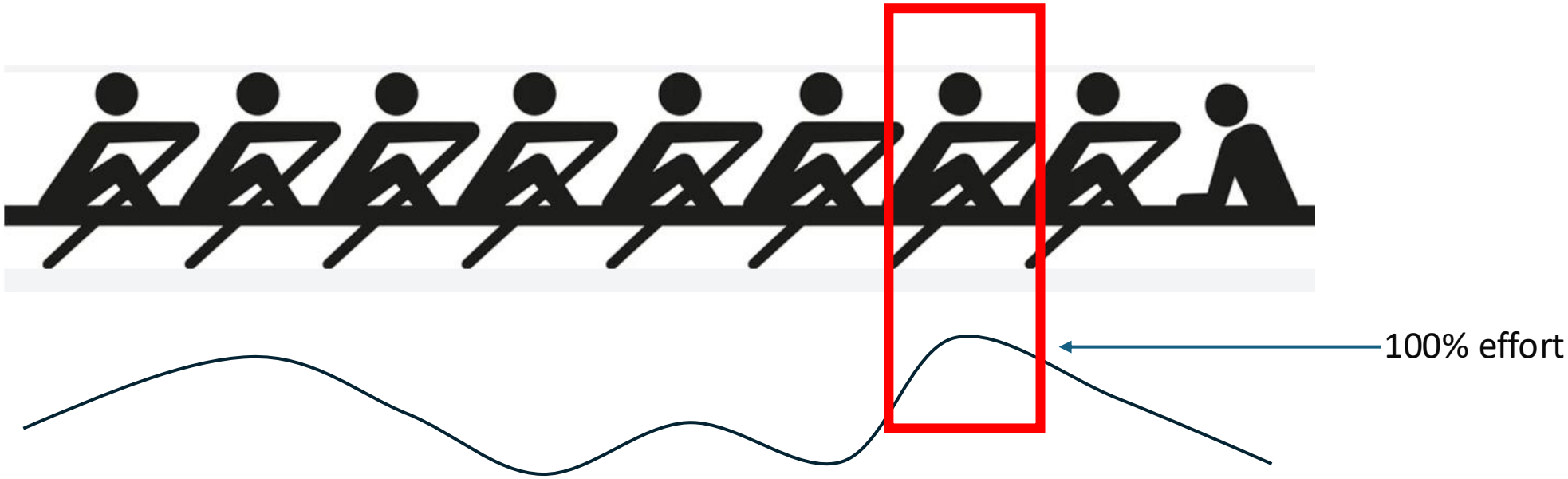
TOC emphasizes that:

improvements should be targeted **solely on the constraint** to achieve quick improvements in productivity and throughput, rather than local optimizations elsewhere

OK – but HOW do I find my constraint?

A simple analogy – where is the constraint?

1. What is the goal of this system?
2. What is the constraint that determines the maximum performance of this system?



Hack: “Finding a Real Constraint” quickly

A quick walk through the ‘hack’

An interactive practical exercise

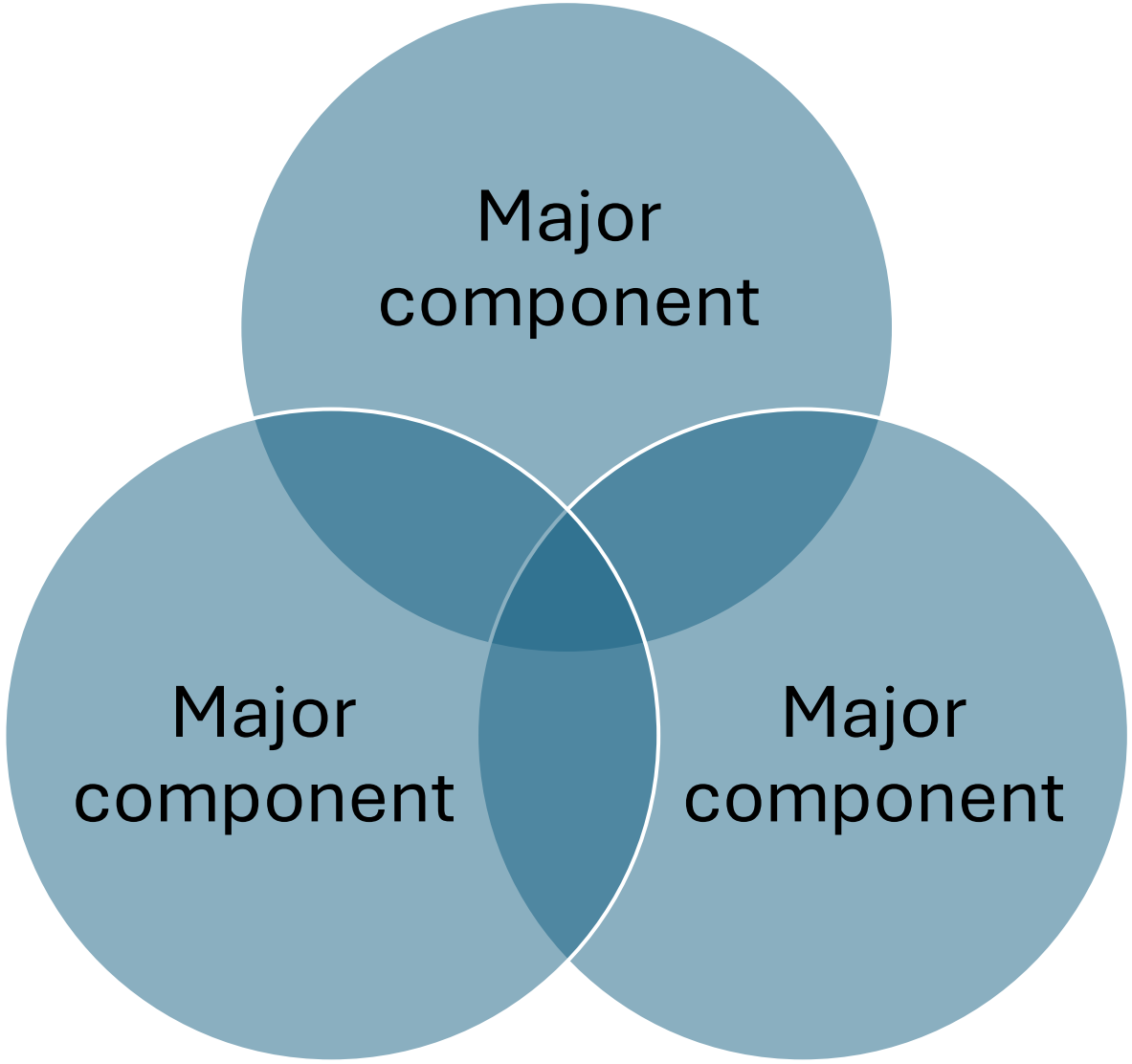
A real-world true example

Exercise – Visualise a System

Q. Which one determines the pace of the whole?

Example:

Design – Build - Integrate



Label that as ***The Constraint***

Exercise – Zoom in (sub-system level)

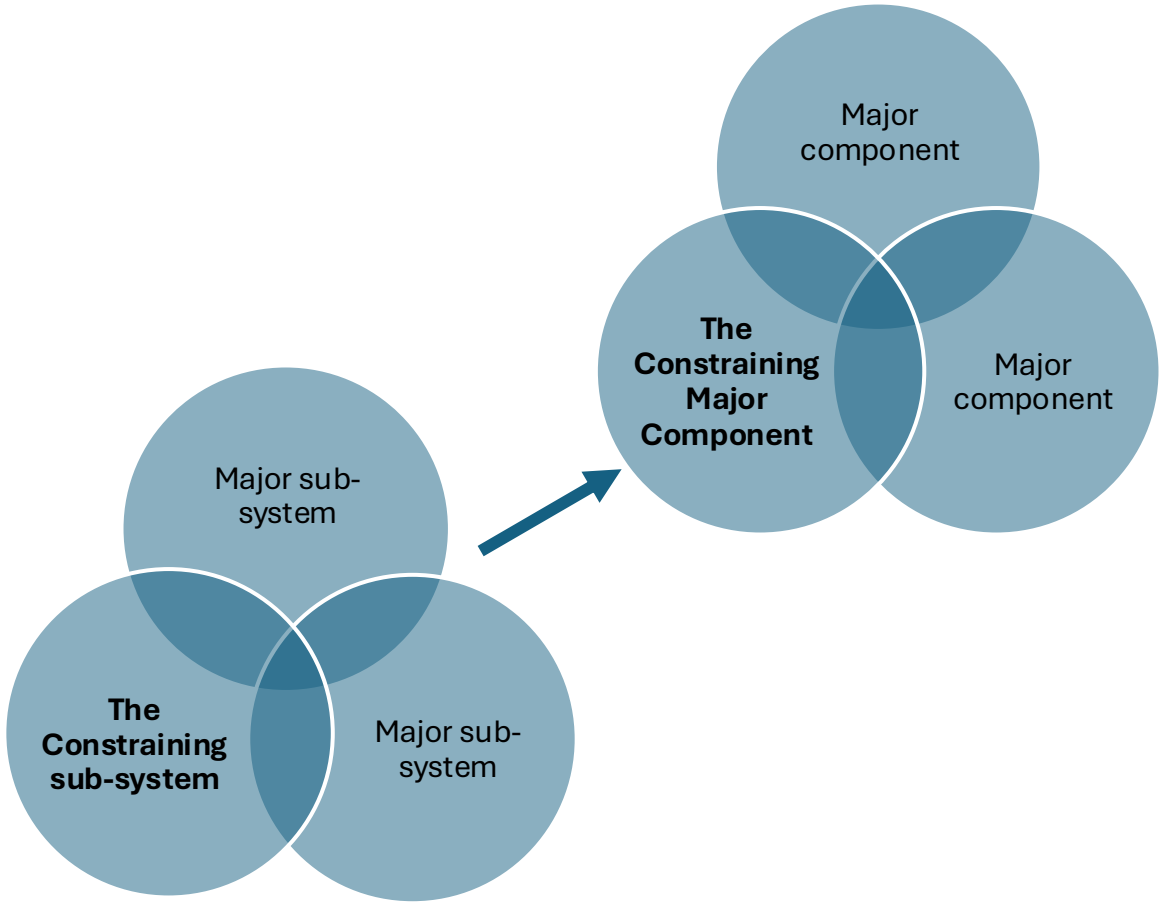
Example:

Planning – Execution - Verification

Ask again:

Which one is limiting

the others?



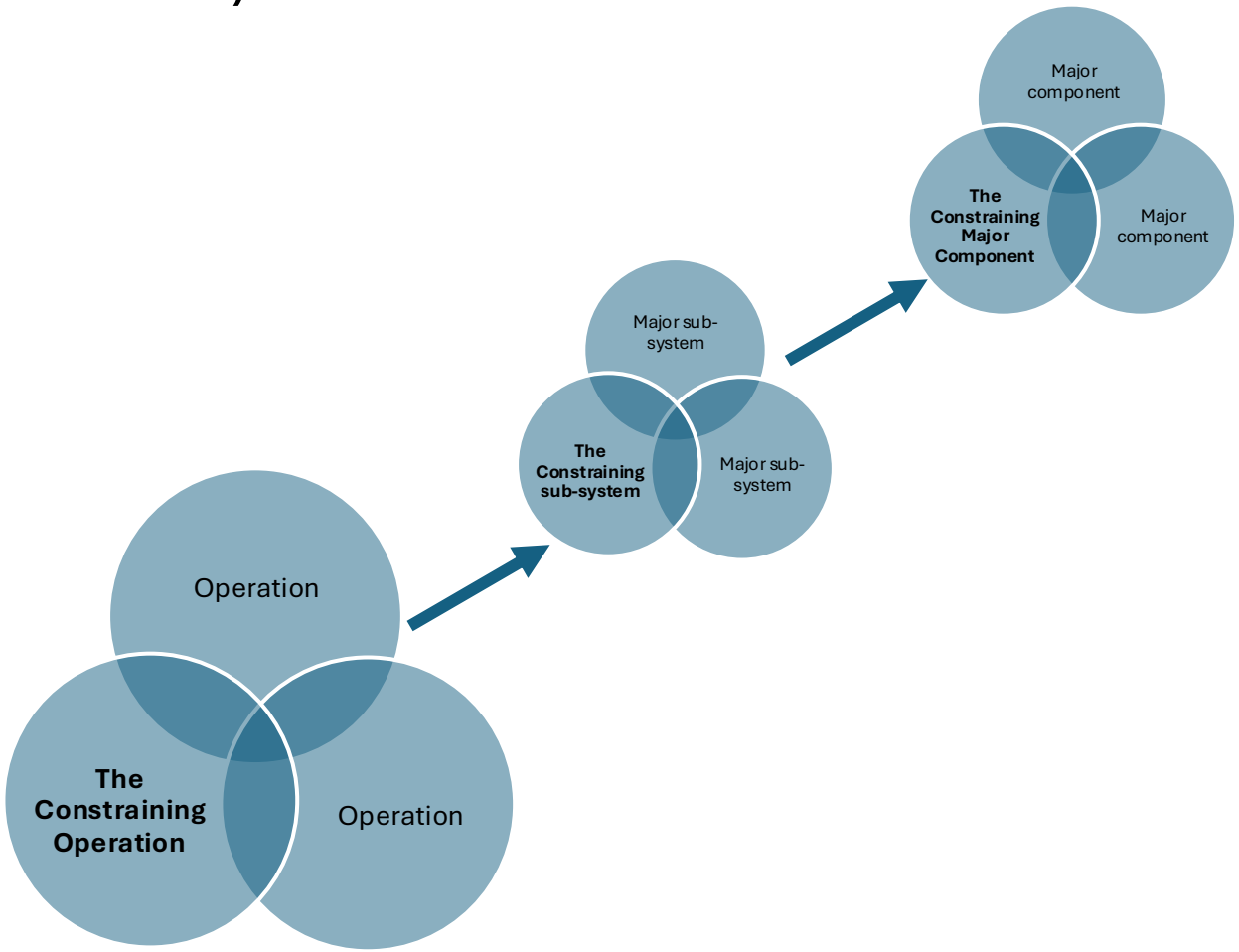
Exercise – Zoom Deeper (Operational level)

Repeat until you land on an

active, observable operation

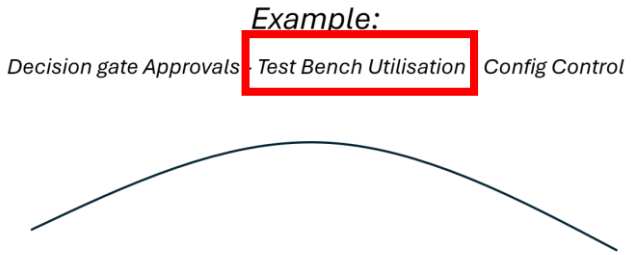
Example:

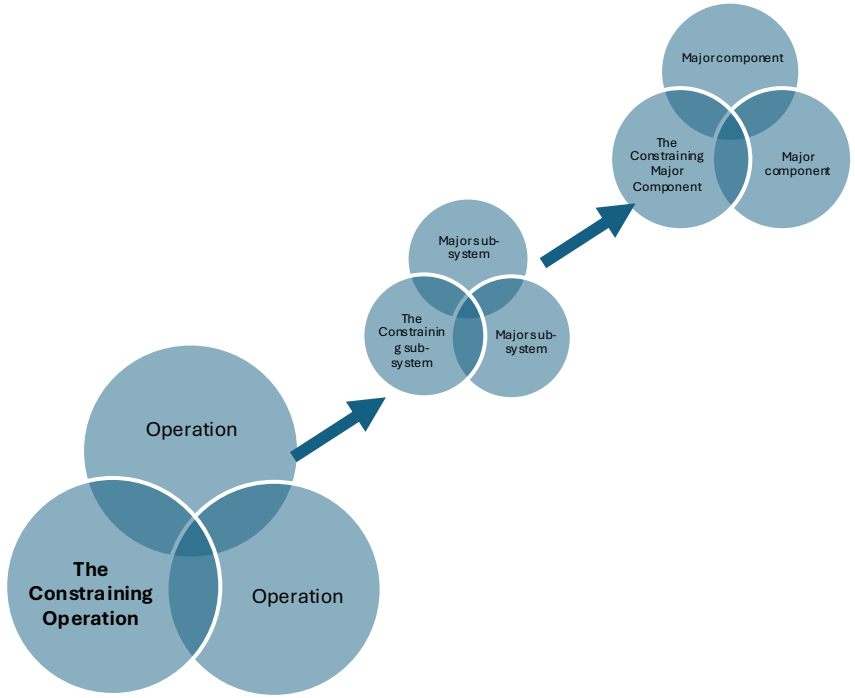
Decision gate Approvals - Test Bench Utilisation - Config Control



Exercise – The Uncomfortable Question

"If this operation's output governs the entire system's throughput, then all improvement must start here."



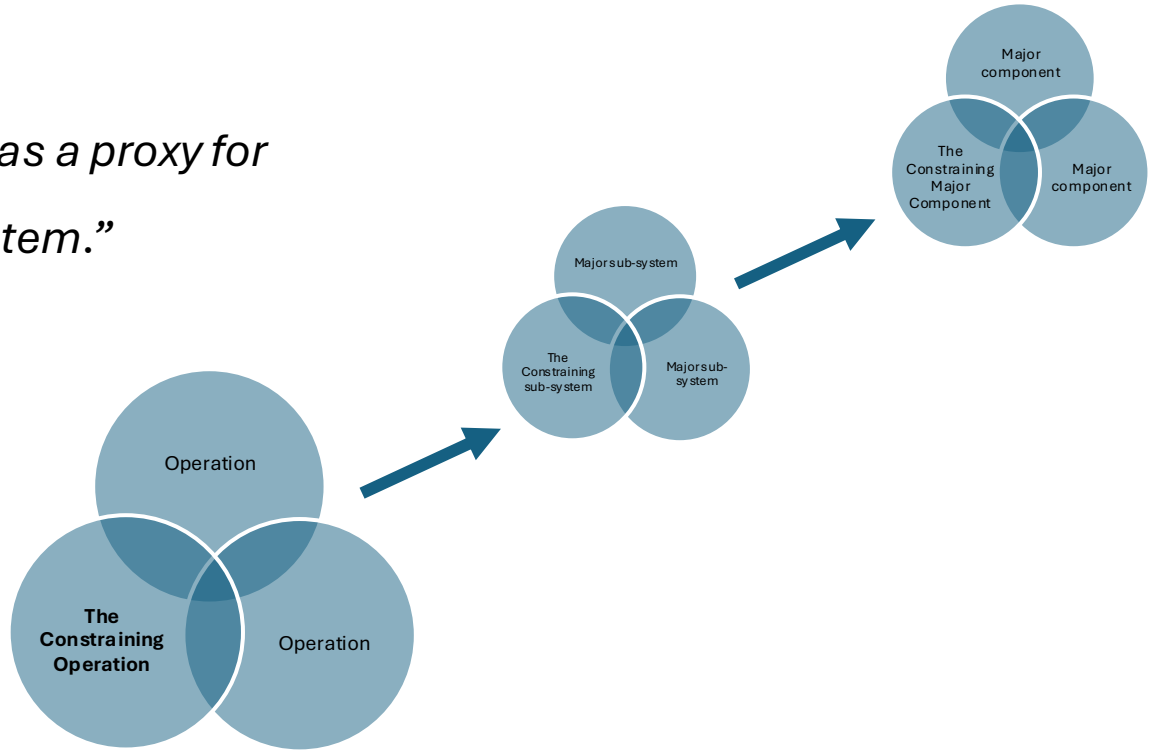


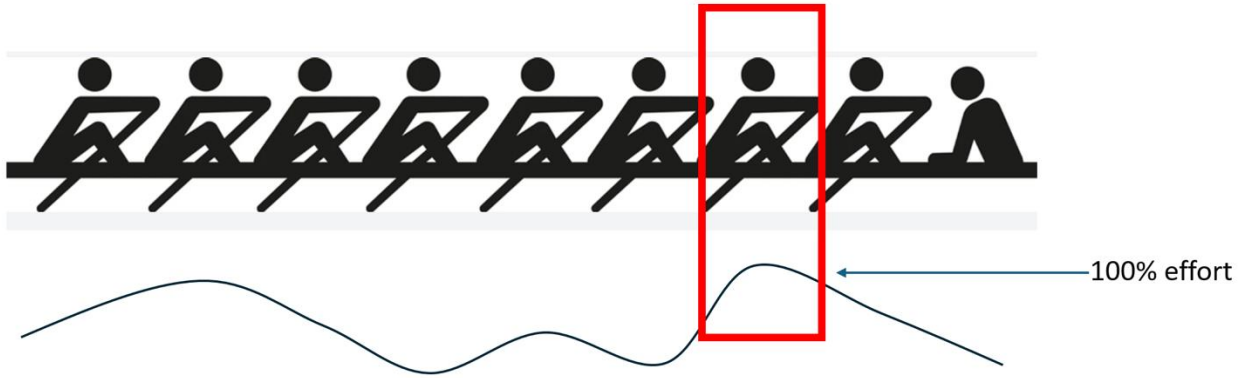
Q. What is the actual reason this operation has flow blocked or choked?

Q. Are these real causes or excuses, masquerading as explanations?

The Reality Check

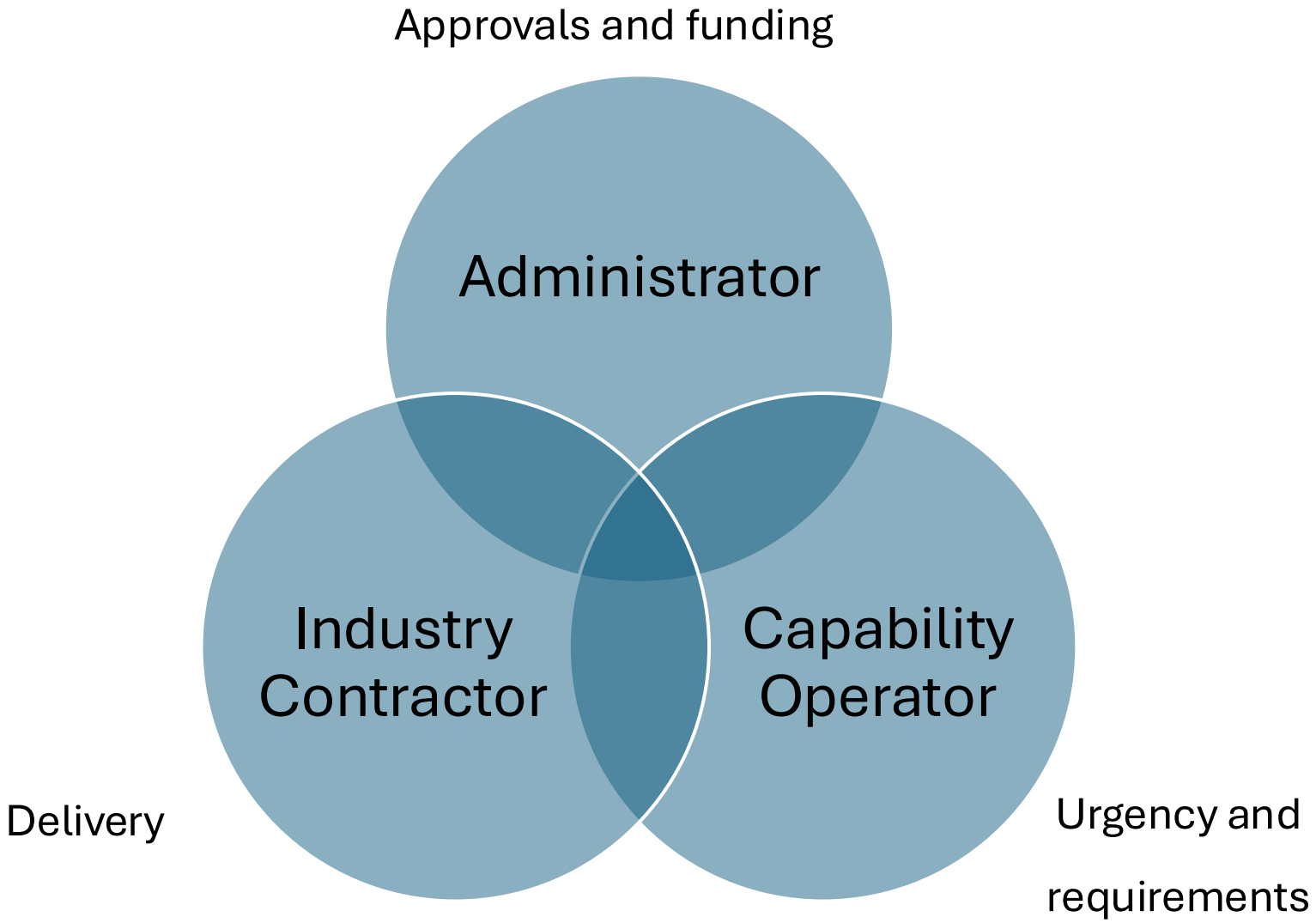
"The performance of the constraint acts as a proxy for the performance of the entire system."





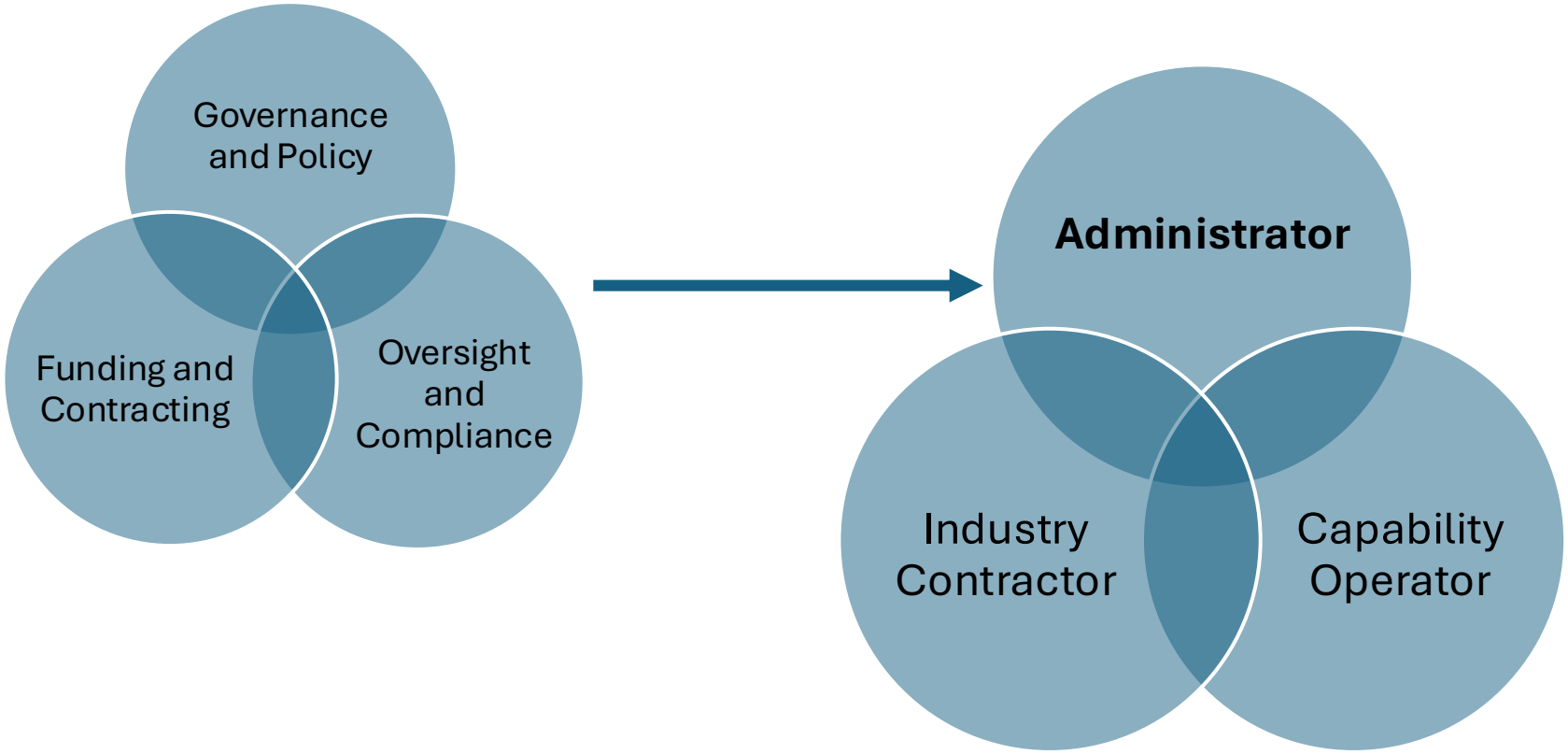
Frame a System – Choose the current constraint

Q. If these three are meant to work together to deliver a capability – which one *actually* determines system performance most of the time?



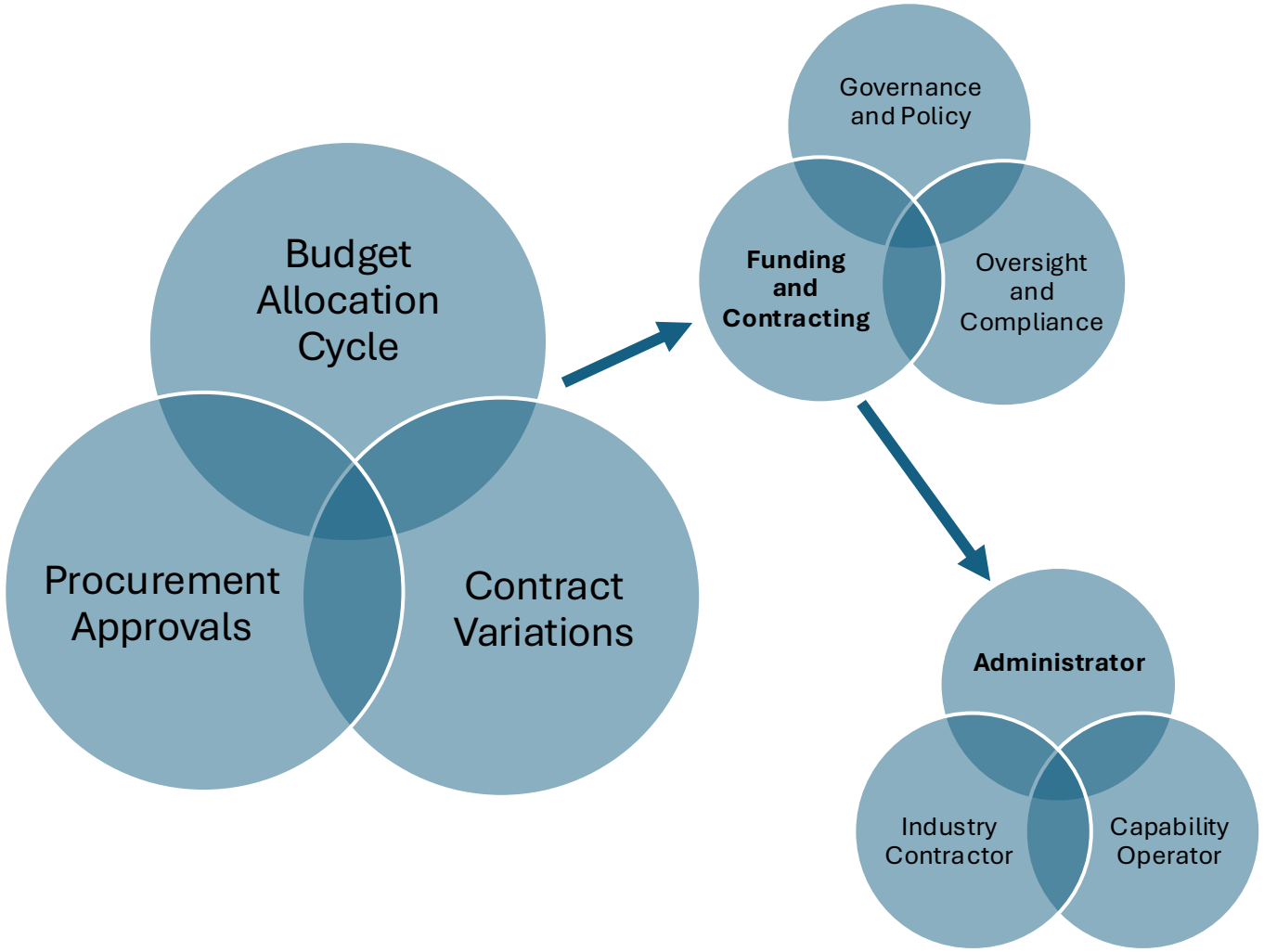
Scenario – Inside The Administrator

Q. Which one of these three most limits how quickly the Administrator can enable capability?



Scenario – Inside Funding and Contracting

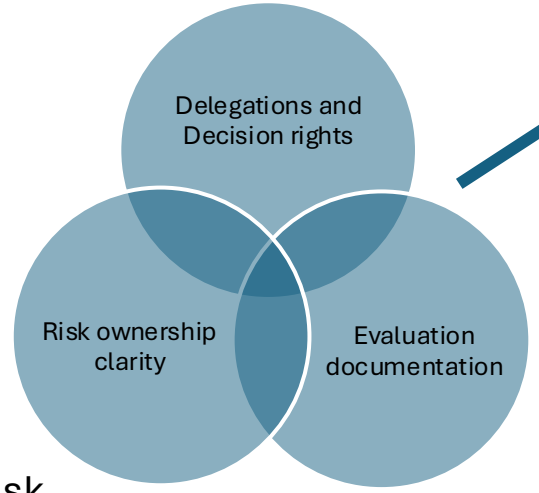
Q. Which one of these constantly throttles the systems ability to flow work?



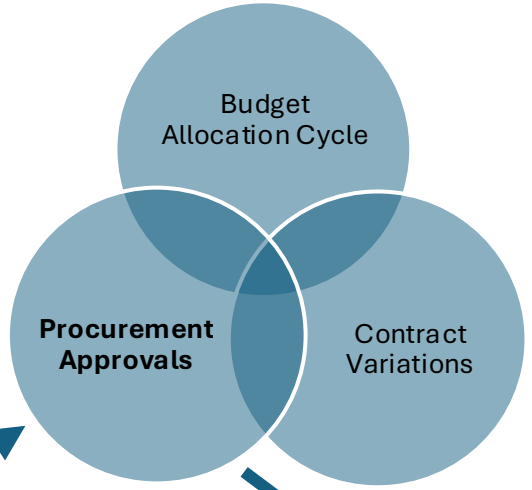
Scenario – Inside Procurement Approvals

This is where it gets real.

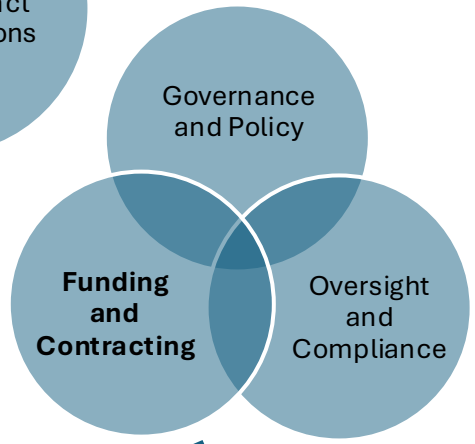
Delegations are unclear

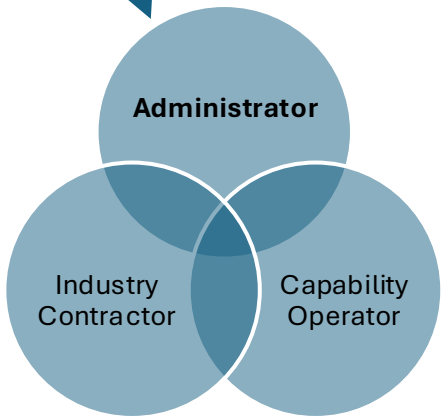


No one wants to own the risk



Waiting on a signature that hasn't been delegated…





Zoom back out – Reflection & Insight

We've reached an *active, operational node:*

The human decision point that throttles the entire capability system.

"If this is idle, *the whole system is idle.*"

When the **Administrator's** decision-making loop stalls,

the **Contractor** waits,

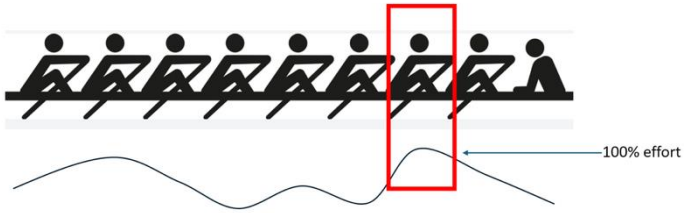
the **Operator** loses tempo,

and *the system's constraint has effectively frozen capability flow.*

Scenario Debrief Thoughts

Constraint ≠ blame.

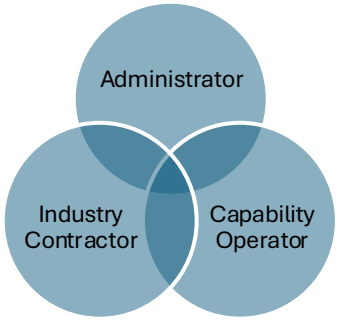
It's just where the system is weakest right now.



The constraint lives where ownership is blurred.

The more shared accountability, the more it hides.

Improving system flow means strengthening constraints together.



Tied back to TOC:

Identify → Exploit → Subordinate → Elevate → **Repeat**.

Quick Dive 2 – The Capability Operator

Core tension: Readiness vs. Flexibility

"If the operator's job is to keep capability available, where does their system slow down?"

Which one most throttles operational readiness?

Top Level (within the Operator)

1. Requirements Definition ←
2. Operational Validation / Test
3. Capability Employment

Typically **Requirements Definition** wins,

because it's slow, ambiguous, and late-changing.

Quick Dive 2 – The Capability Operator

Top Level (within the Operator)

1.Requirements Definition

2.Operational Validation / Test

3.Capability Employment

Inside Requirements Definition

1.Doctrine alignment

2.User Needs articulation

3.Change management process

Usually **User Needs articulation**, because the operator community speaks in effects, not specifications

Where do things really stall?

Quick Dive 2 – The Capability Operator

Top Level (within the Operator)

1.Requirements Definition

2.Operational Validation / Test

3.Capability Employment

Inside Requirements Definition

1.Doctrine alignment

2.User Needs articulation

3.Change management process

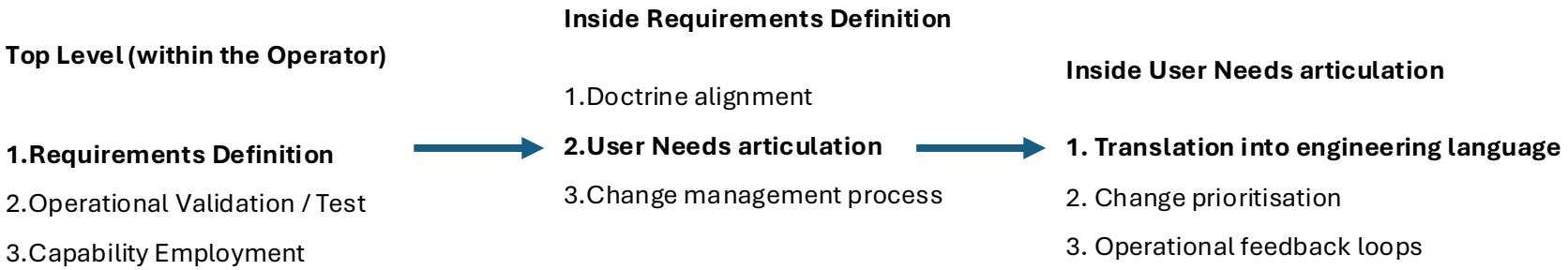
Inside User Needs articulation

1. Translation into engineering language
2. Change prioritisation
3. Operational feedback loops

Why?

*Because it sits right at the interface with industry – where **intent meets implementation***

Systems Thinking Insight – The Capability Operator



Question:

If not being able to translate requirements/intent into engineering language was a constraint,

that acted as a proxy for the performance of the whole system,

how could the Administrator and the Industry Contractor remove that constraint?

Real Quick Dive 3 – The Industry Contractor

Core tension: Speed vs Quality

"If the contractor is accountable for output, where does their throughput really get throttled?"

Where does intuition suggest the constraint most likely is?

Top Level

1. Design and Development
2. Production/Integration ←
3. Support and Sustainment

Typically in **Integration**, especially in multi-tier defence supply chains

Real Quick Dive 3 – The Industry Contractor

Top Level

1.Design and Development

2.Production/Integration

3.Support and Sustainment

Inside Integration

1.Interface Management

2.Subcontractor readiness

3.Test and Acceptance

Constraint: Interface Management

This is where conflicting standards, late data and change requests typically accumulate

Real Quick Dive 3 – The Industry Contractor

Top Level

1.Design and Development

2.Production/Integration

3.Support and Sustainment

Inside Integration

1.Interface Management

2.Subcontractor readiness

3.Test and Acceptance

Inside Interface Management

1.Technical data exchange

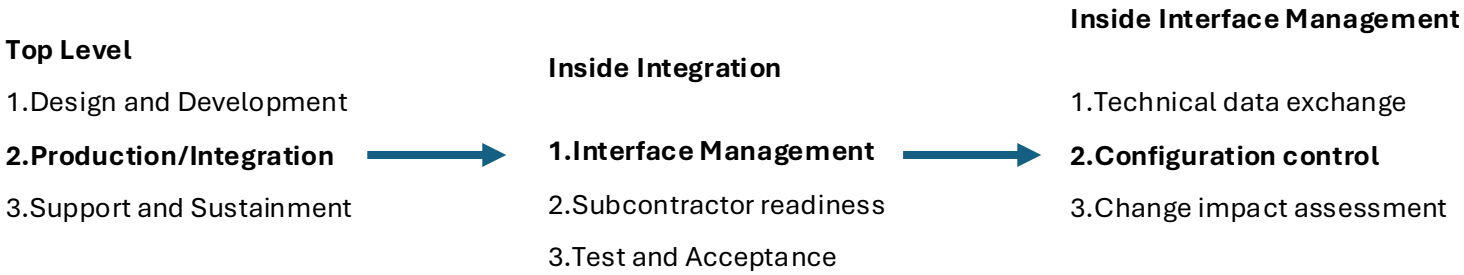
2.Configuration control

3.Change impact assessment

Constraint Revealed: Configuration Control

Often frozen to prevent chaos – yet the freeze itself causes chaos right when the system needs agility …

Systems Thinking Insight – The Industry Contractor



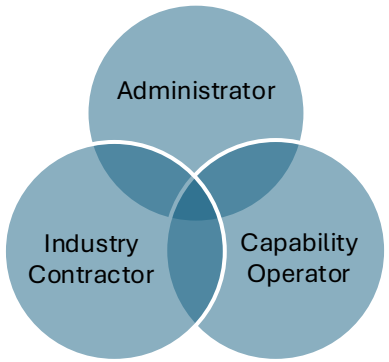
Question:

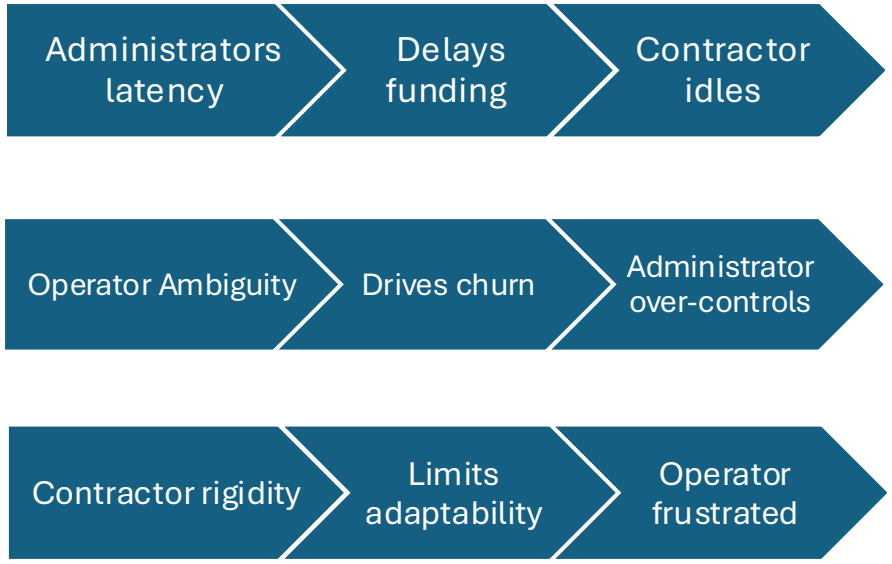
If experiencing conflict from adaptability vs configuration control was a constraint,

that acted as a proxy for the performance of the whole system,

how could the Administrator and the Capability Operator remove that constraint?

Synthesis – Seeing the Whole System





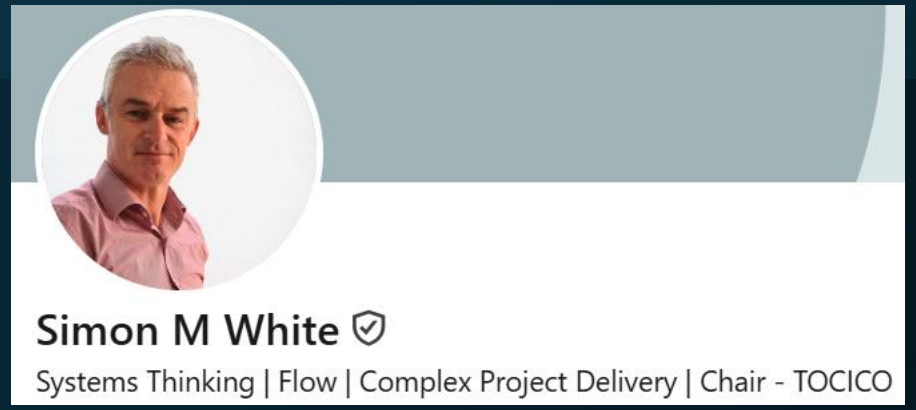
A real example

How Station ‘X43’ acted as a proxy for billions of value per annum, approx. $425,000 per hour

Thank you.
Questions?

Simon M White | LinkedIn

www.simonmwhite.au



Appendix

Systems Thinking Insight – The Capability Operator

Top Level (within the Operator)

1.Requirements Definition

2.Operational Validation / Test

3.Capability Employment

→

Inside Requirements Definition

1.Doctrine alignment

2.User Needs articulation

3.Change management process

→

Inside User Needs articulation

1. Translation into engineering language

2. Change prioritisation

3. Operational feedback loops

"The Operator's constraint is not their capability — it's their clarity."

→

Intent isn't translatable into contractual or technical terms early enough, leading to "requirements churn."

System leverage:

Introduce *boundary translators* — people who can speak both "warfighter" and "engineer."

Freeze intent earlier; let evolution occur downstream through modular design.

Systems Thinking Insight – The Industry Contractor

Top Level

1.Design and Development

2.Production/Integration

3.Support and Sustainment

→

Inside Integration

1.Interface Management

2.Subcontractor readiness

3.Test and Acceptance

→

Inside Interface Management

1.Technical data exchange

2.Configuration control

3.Change impact assessment

"The Contractor's constraint is not their capacity — it's adaptability."

→

They can build fast, but not safely adapt to shifting external dependencies.

System leverage:

Move from waterfall acceptance to continuous verification.

Co-locate engineers with operators earlier in the cycle.

Use integrated digital twins to reduce rework latency.