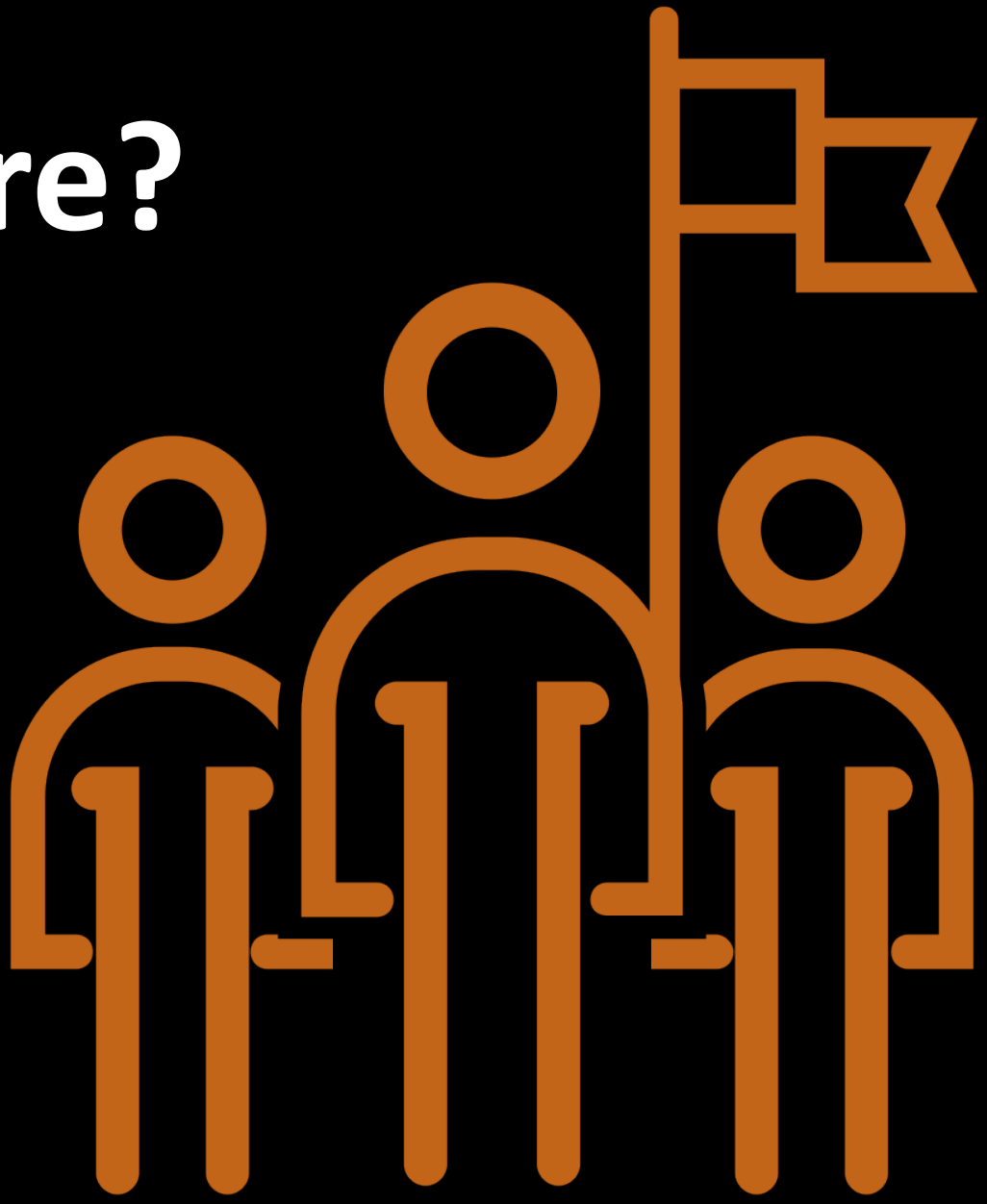
The Emperor's *old* clothes
Engineering Culture
Revived

Finbarr Joy

Culture?



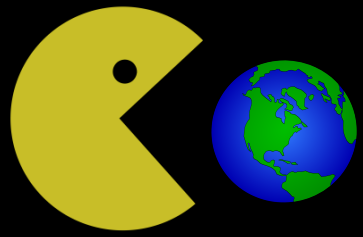
A pattern of basic assumptions that the group has invented, discovered, or developed in learning to cope with its problems of external adaptation and internal integration.

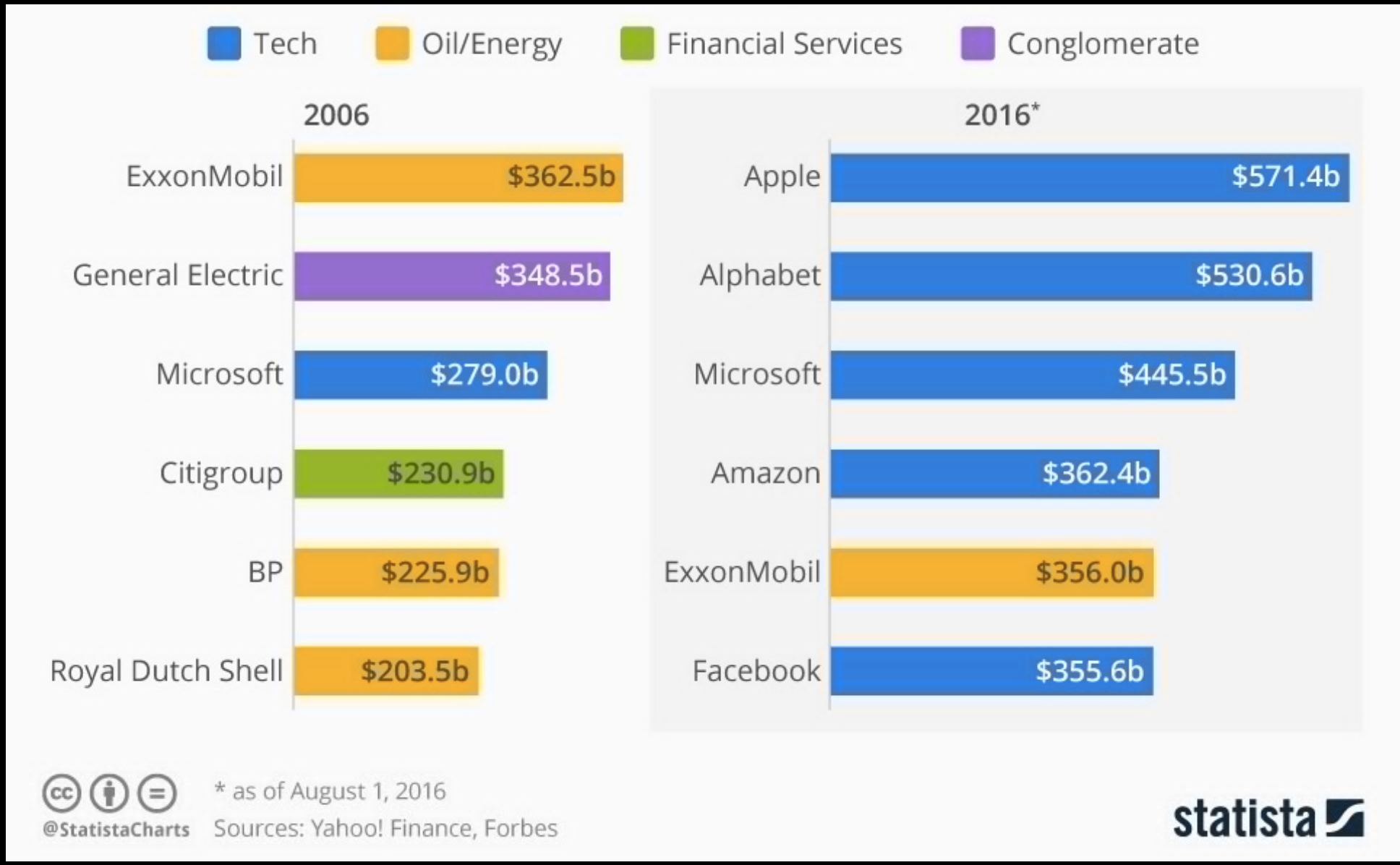
Edgar Schein

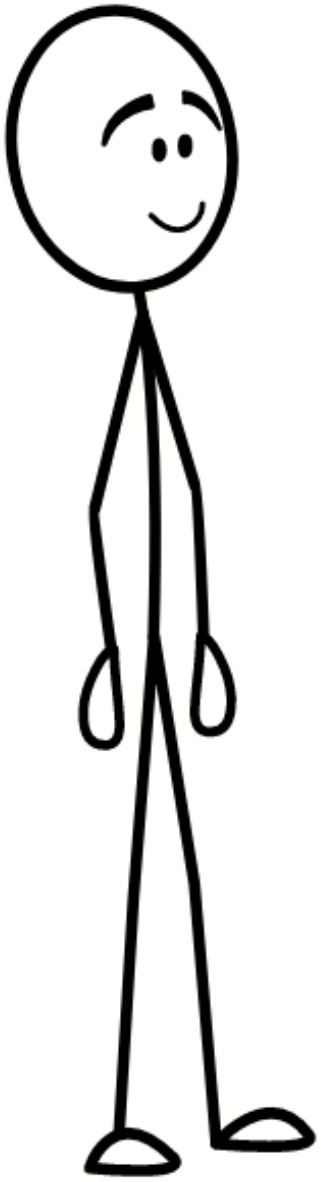
*sharpening the organisation's **instinct**:*

- how we compete
- how we deal with challenges
- how we relate to one another
- how we agree the 'right' thing to do
- how we *behave*
- how we *react*

What's going *on*?



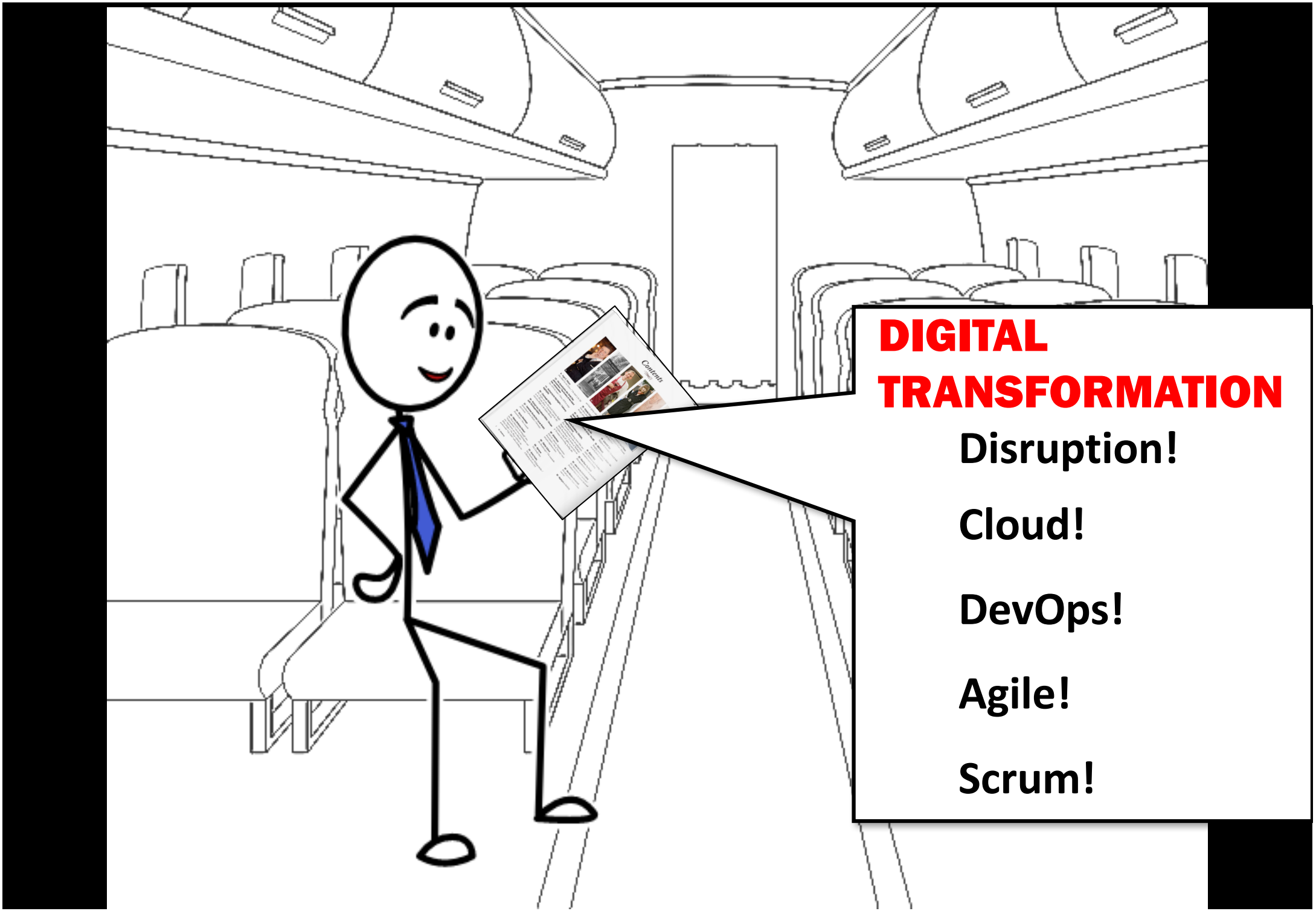


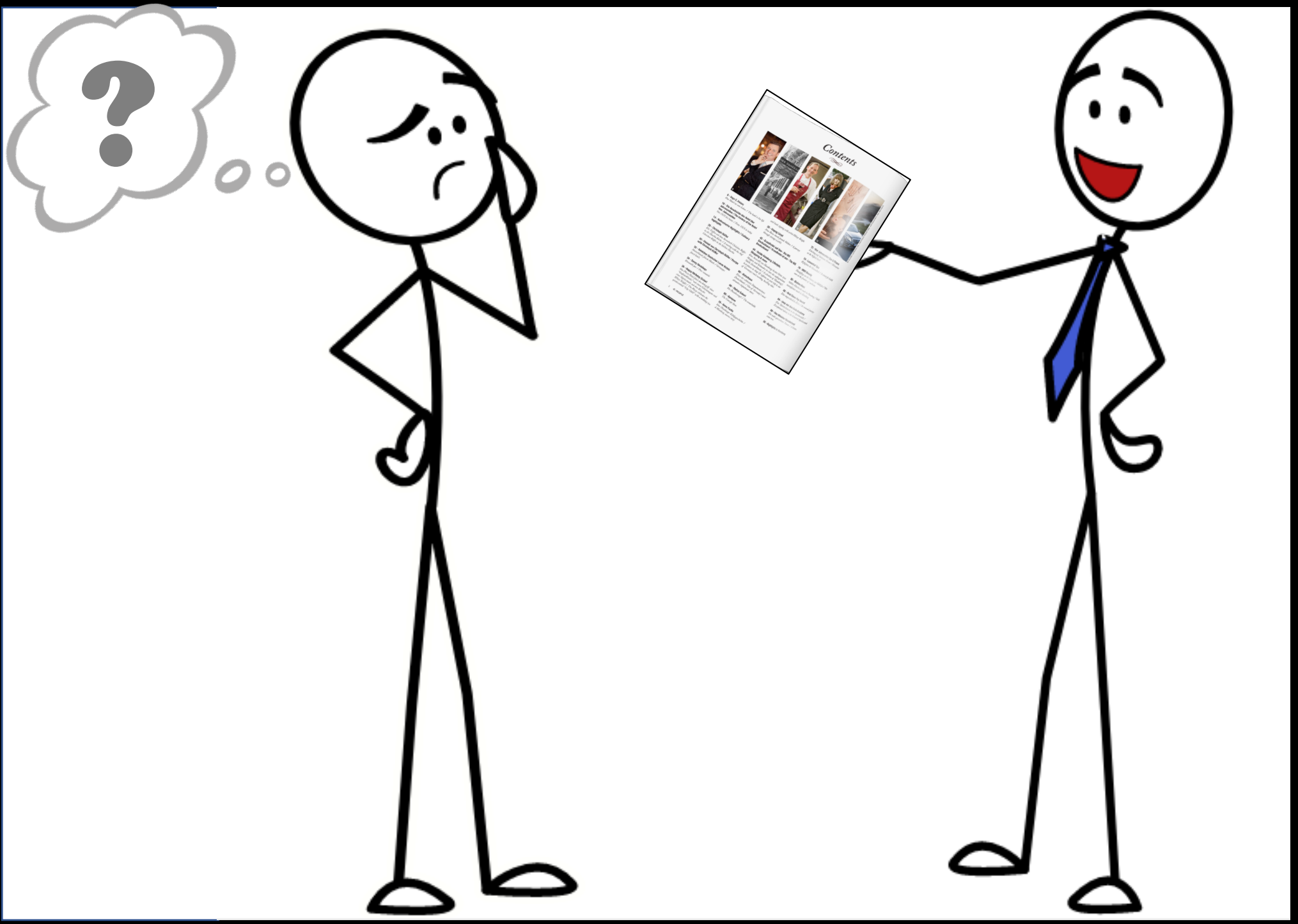


React..? Vue..?

Go..? Rust..?

home..? pub..?


DIGITAL TRANSFORMATION
Disruption!
Cloud!
DevOps!
Agile!
Scrum!
Contents


?
Contents

Digital Transformation

Be Like Steve !

Culture Transformation is key..

for when “employees balk at the new practices required by the new technologies.”

the best way to establish a culture?

DON'T *'establish' a culture !!*

Walk the walk

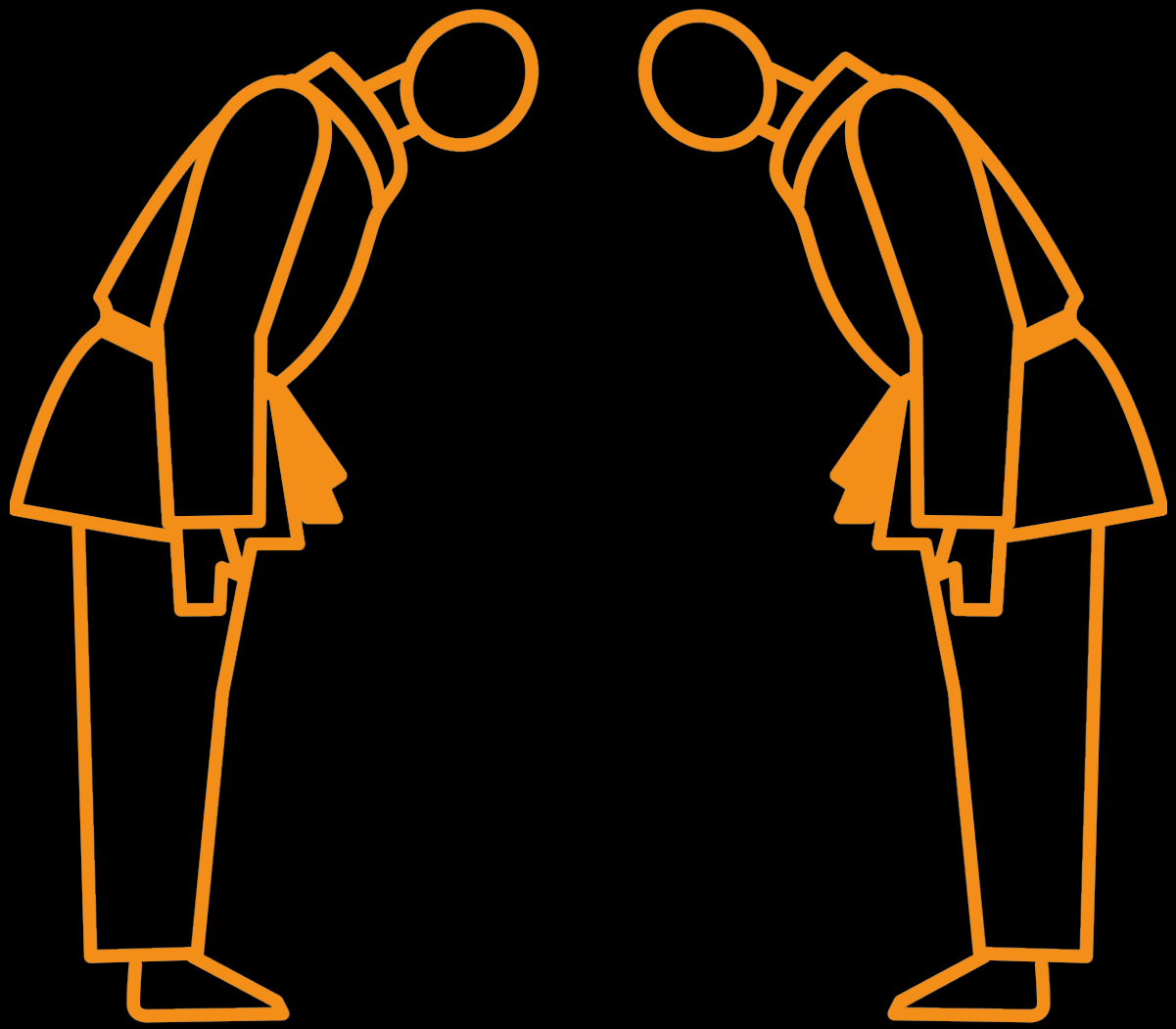
https://www.ozprinciple.com

Leader? Be. Do. Say

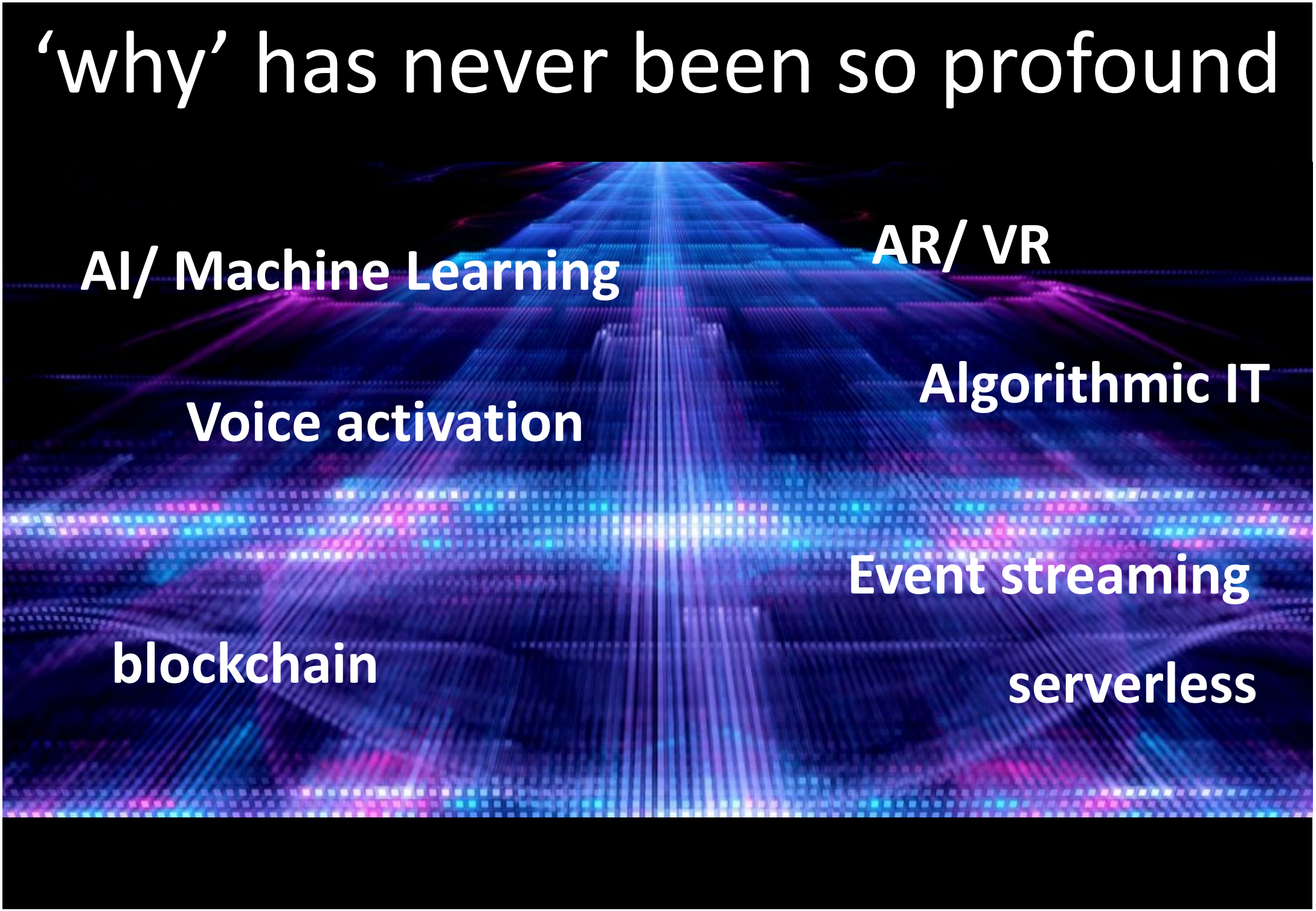
- What your focus is – what you're measuring
- How you react to incidents / crises
- How you allocate resources
- How you allocate rewards & status
- How you recruit/ promote/ select

Role Modelling, teaching and coaching

respect



Start with a *mission*


‘why’ has never been so profound
AI/ Machine Learning
AR/ VR
Algorithmic IT
Voice activation
Event streaming
blockchain
serverless

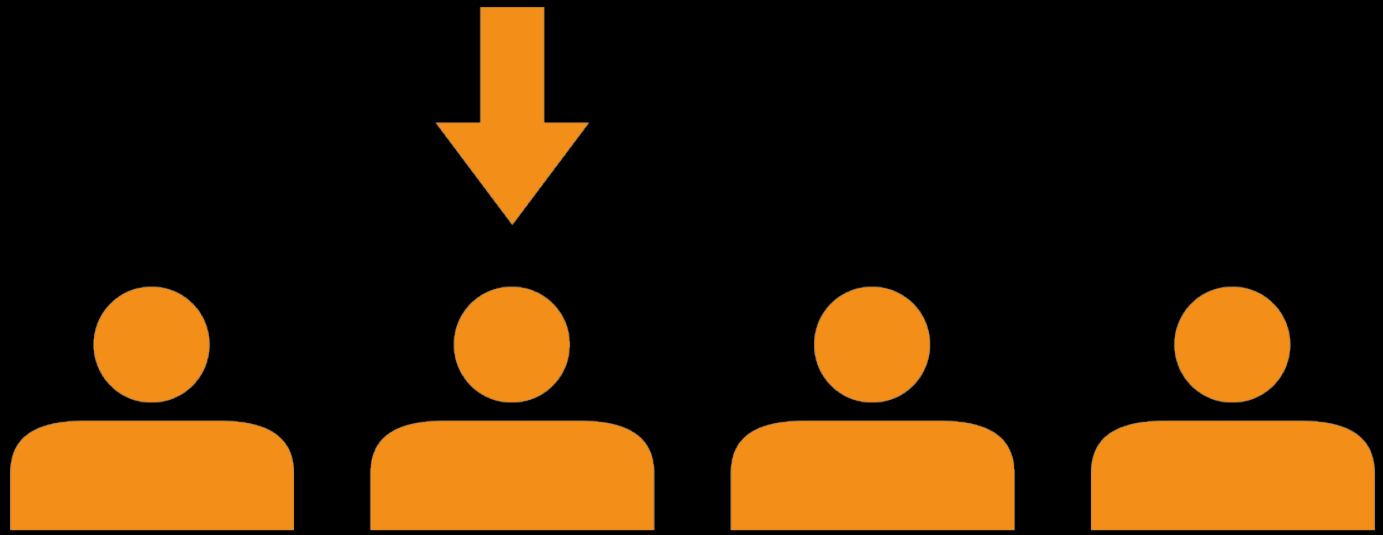
INNOVATION LAB?

if you’re not innovating why write software ?

MARKETING TECHNOLOGY LUMAscape

The ‘right’ team

selection?



"You can't have great software without a great team, and most software teams behave like dysfunctional families."

- Jim McCarthy

How we work

Religion is regarded by the common people as true, by the wise as false, and by rulers as useful.

Lucius Annaeus Seneca

How we work

Agile is regarded by the common people as true, by the wise as false, and by rulers as useful.

“Four measly bullets, and all this s#!t happened?!“

Jon Kern

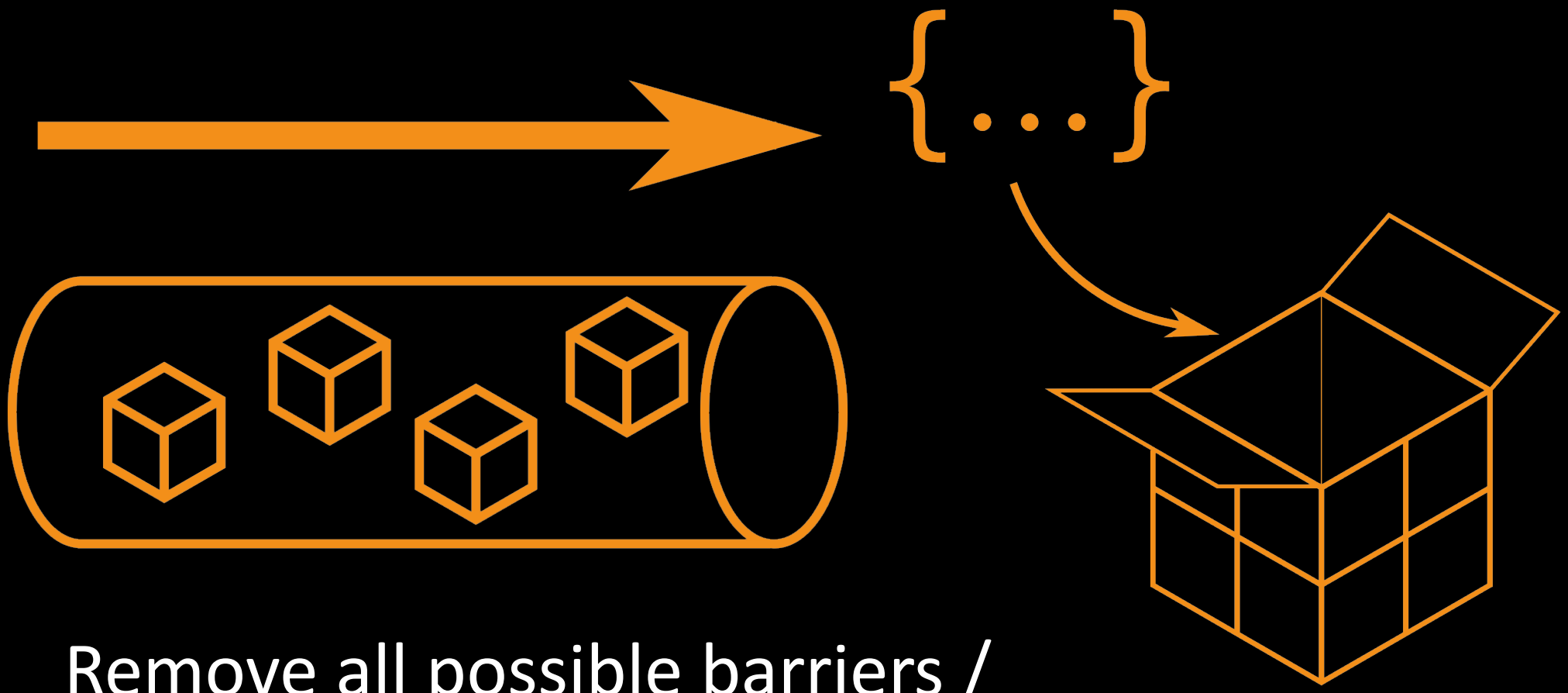
Contino - *Secrets from the Agile Manifesto Authors on Flow*


is it SAFe?

Pattern of work: *flow*

shortest possible time to release

Pattern of work: *flow*



Remove all possible barriers /
dependencies to release: *automation*

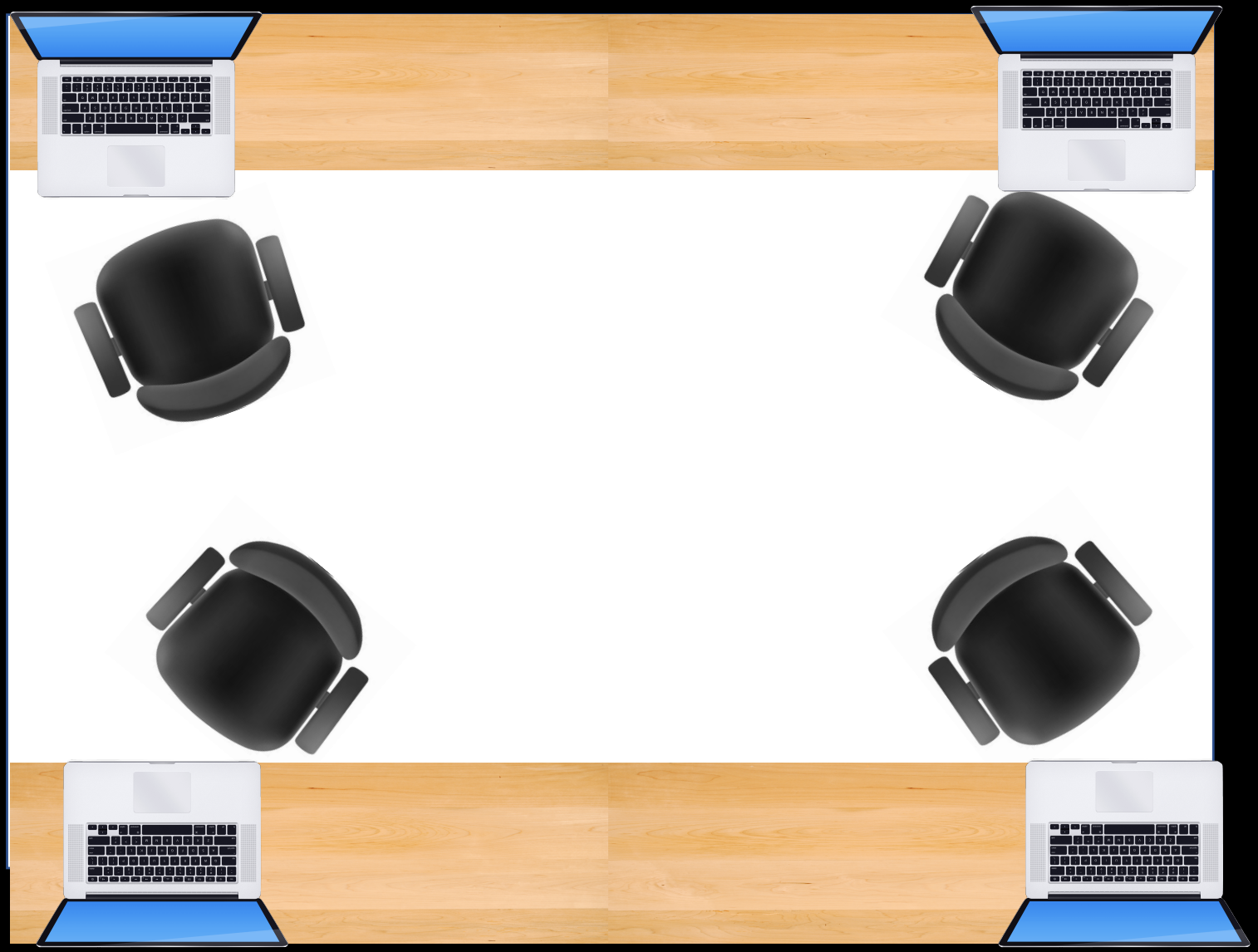
CODE MAINTENANCE

The price of reliability is the pursuit of the utmost simplicity. It is a price which the very rich find most hard to pay

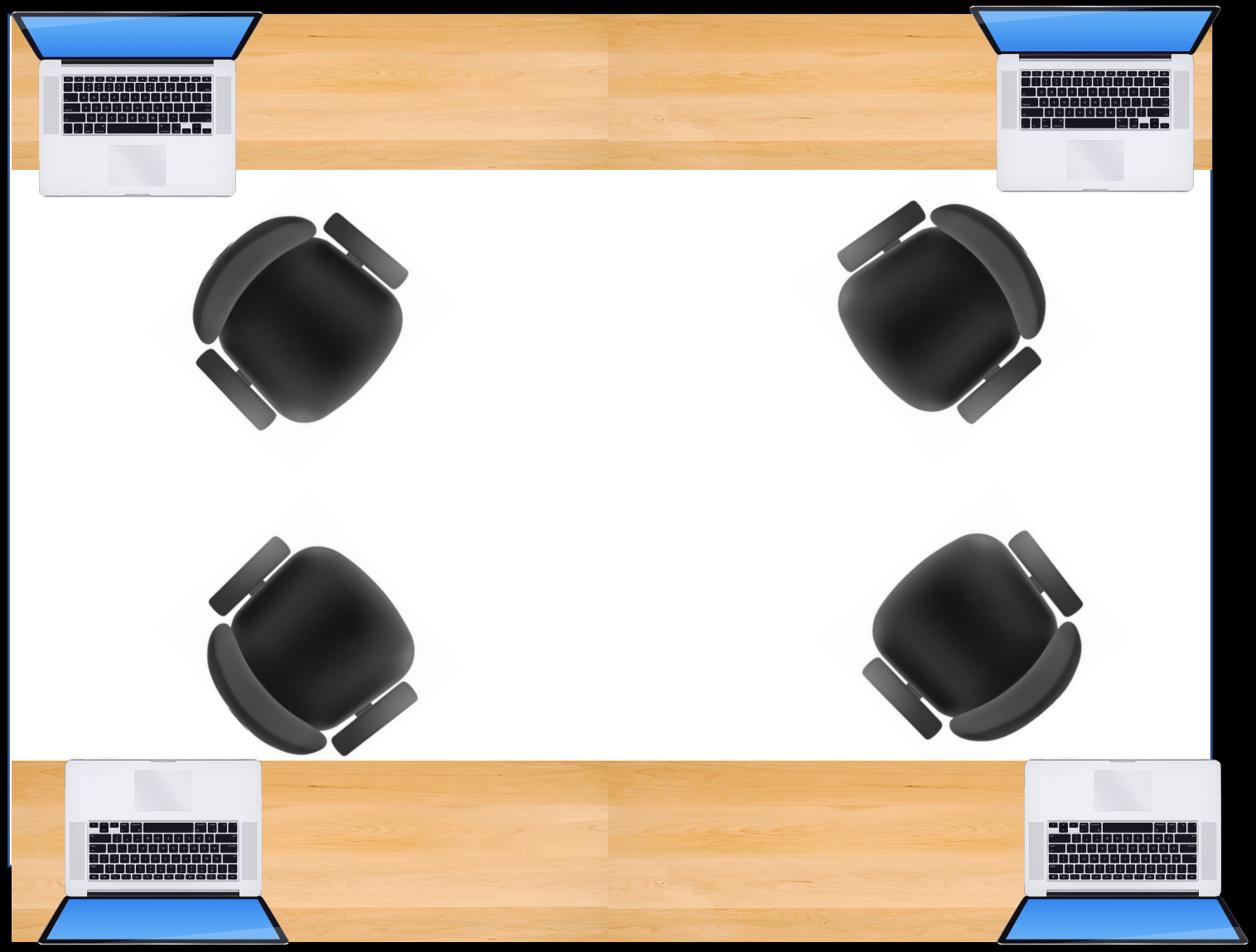
C.A.R Hoare

Environment ?

Environment ?



Environment ?

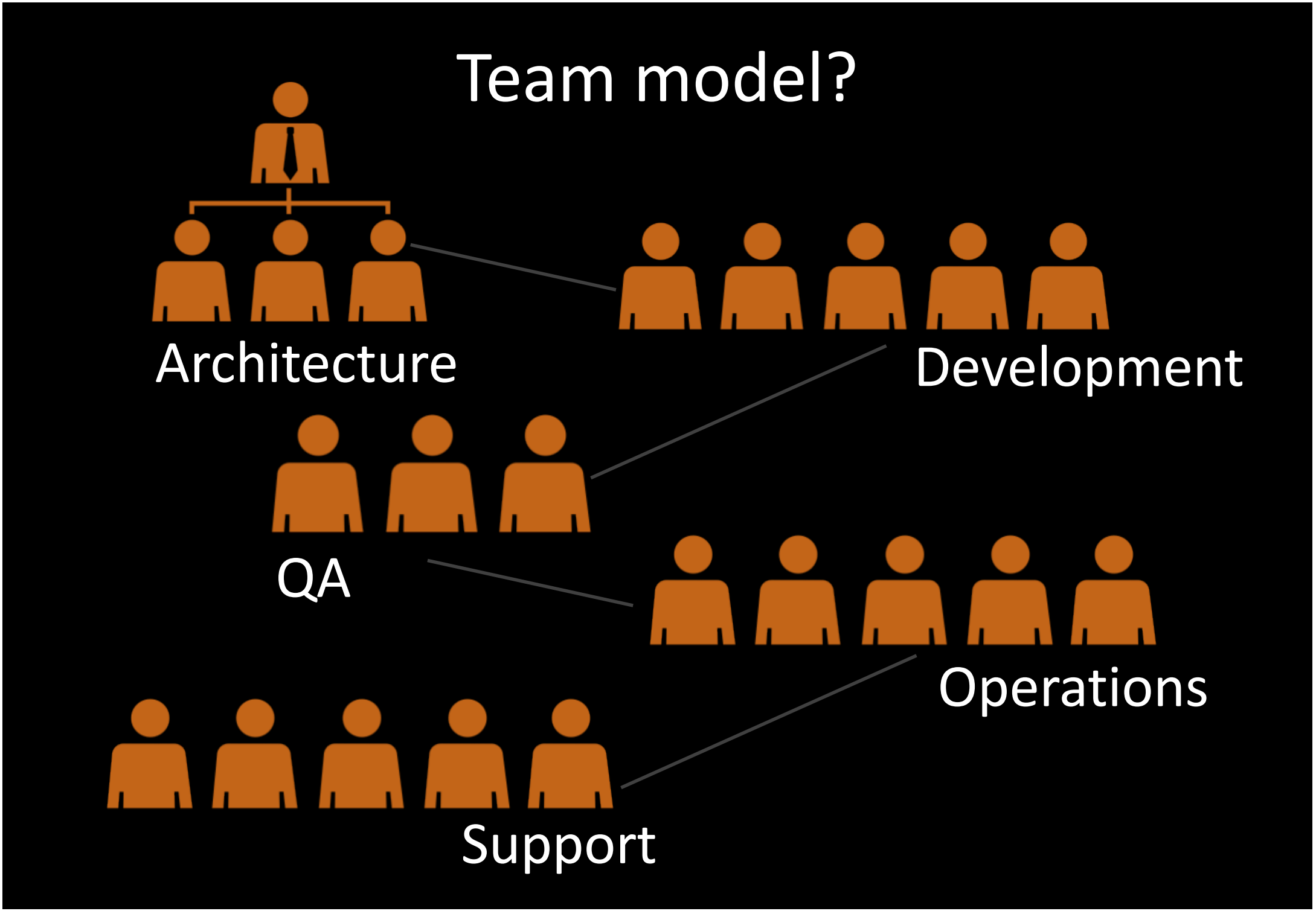


Must have ‘enough’ autonomy

- Leadership clear on ‘why’ & ‘what'
- tools, technologies and HOW
- Developer evaluates best options for release
- Developer releases to live
- *Invested in* the customer lifecycle

walk the walk: ’just’ programming?

walk the walk: ’just’ programming?


Team model?
Architecture
Development
QA
Operations
Support

Team model?

Architecture

Product Management

Development

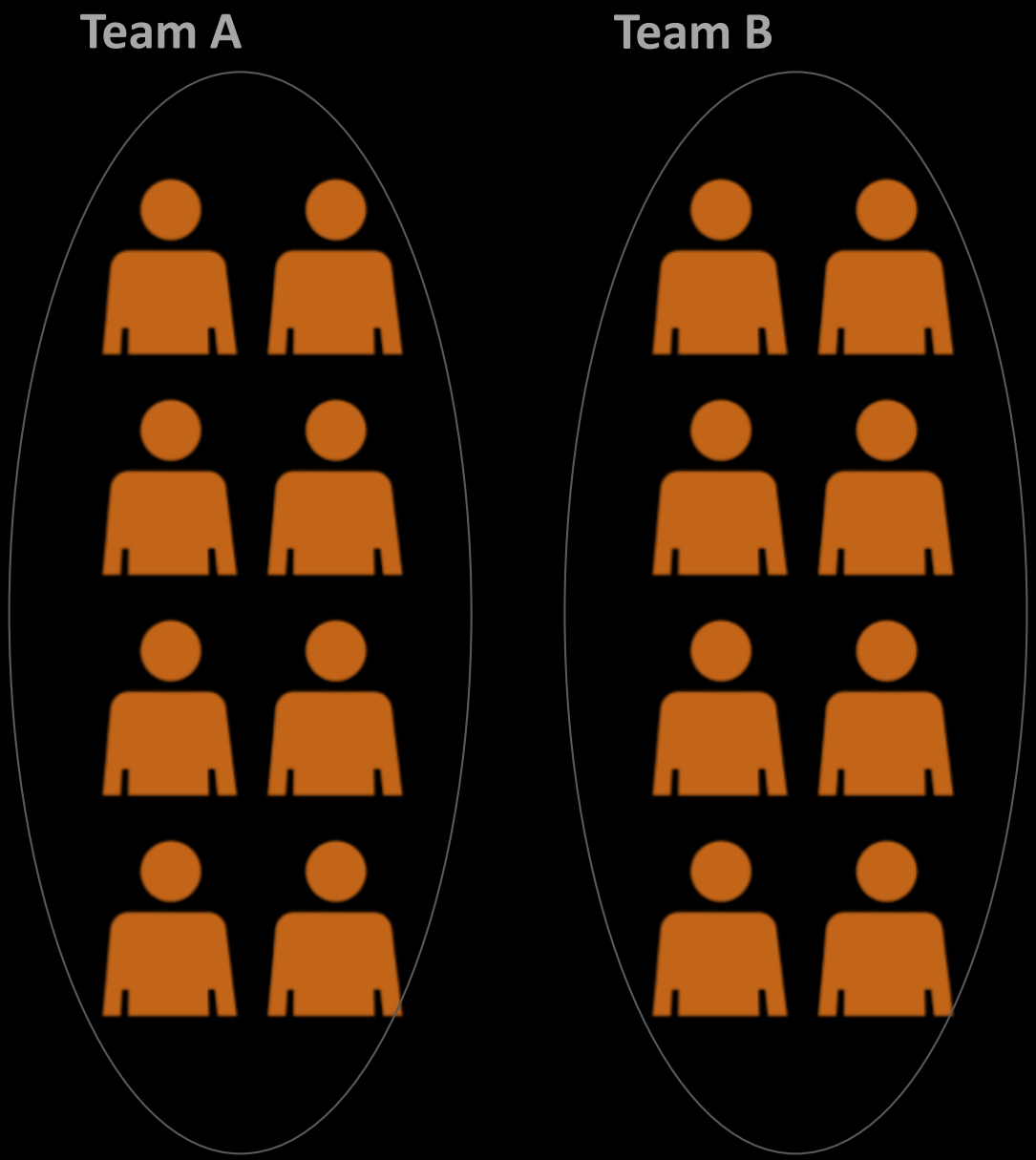
CX

QA

Mkt Analytics

Operations

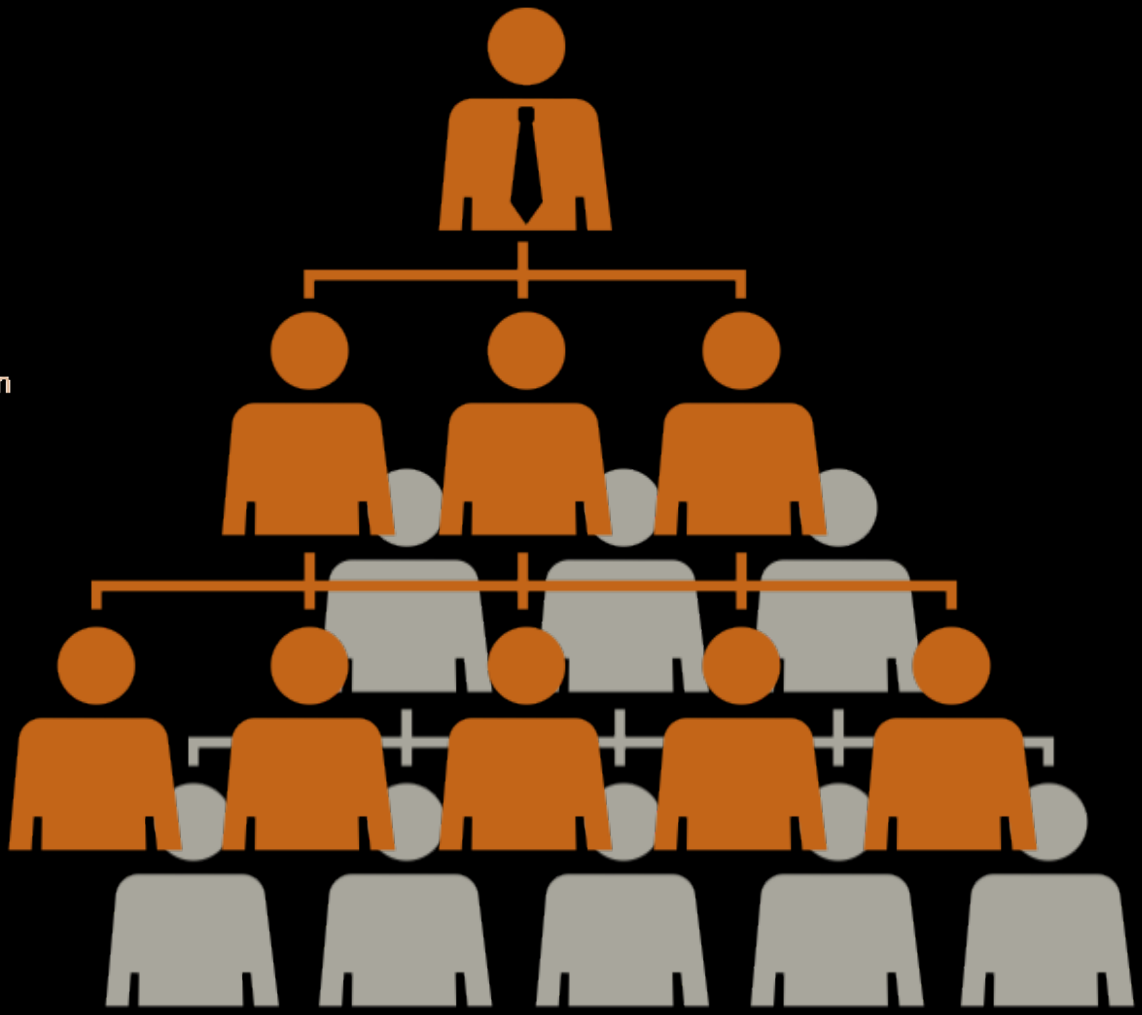
Support



Structure

$$F_{net\ external} = ma$$

Net force on object = mass of object x acceleration



Structure

From teams, build communities

- Walk the walk on your values
- Create a mission
- Embed Innovation
- (Self)Select the best team members
- Appreciate the developer ‘psyche’
- Consider the ‘rhythm’ of release (avoid deathmarch)
- Remove all impediments (bureacracy/ automation)
- Stay on top of the ‘housekeeping’
- Environment sustains communication
- Don’t accept ‘proxies’ to customer
- Re-imagine your team/ org shape – Network!
- **Re-claim engineering principles !**

Momentum is everything

- Make each service **do one thing well**. To do a new job, **build afresh** rather than complicate existing services by adding new features.
- Expect the **output** of every service to become the **input** to another, as yet unknown, service..
- Design and build software to be **tried early**, ideally **within weeks**. Don’t hesitate to throw away the clumsy parts and rebuild them.
- Use tools …to lighten an engineering task, even if you have to detour to build the tools and expect to throw some of them out after you’ve finished using them.

- Make each service **do one thing well**. To do a new job, **build afresh** rather than complicate existing services by adding new features.

cloud/ micro-services

- Expect the **output** of every service to become the **input** to another, as yet unknown, service..
- Design and build software to be **tried early**, ideally **within weeks**. Don't hesitate to throw away the clumsy parts and rebuild them.

agile

- Use tools in preference to unskilled help to lighten an engineering task, even if you have to detour to build the tools and expect to throw some of them out after you've finished using them.

DevOps?

only slightly modified from: "Bell Labs' Unix Timesharing Systems" Documentation 1978: https://ia902701.us.archive.org/12/items/bstj57-6-1899/bstj57-6-1899.pdf

we're hiring !

Questions

@FinbarrJoy

finbarr.joy@gmail.com

linkedin.com/in/finbarrjoy