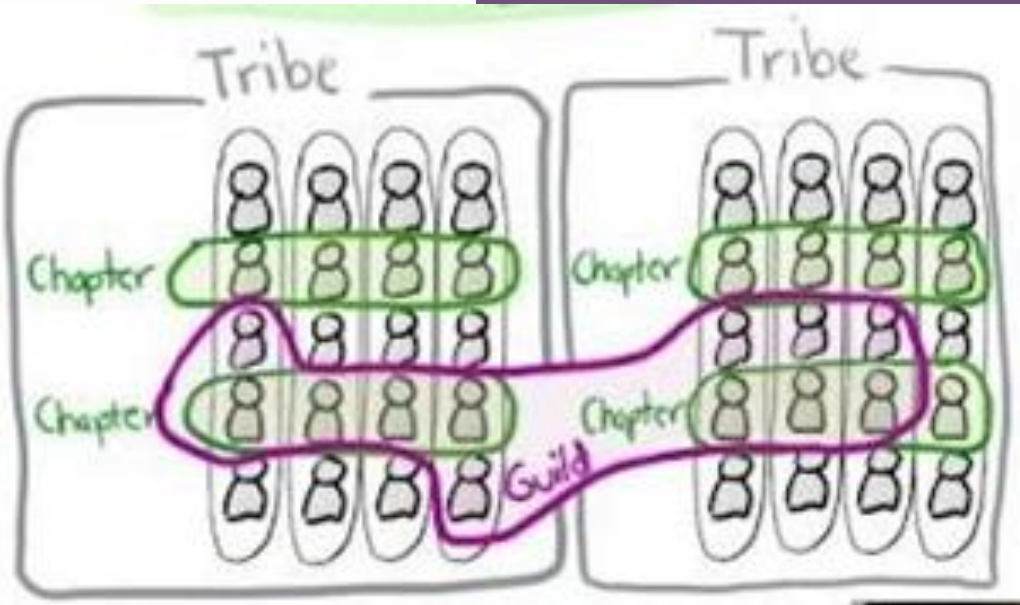
BUSINESS AGILITY INSTITUTE

Spotify®

The top 3 points you should have paid attention to in the Spotify Engineering Culture videos that aren't Squads, Chapters, Tribes, Guilds

Jason Yip
Staff Agile Coach, Spotify Advertising
@jchyip
https://jchyip.medium.com
jyip@spotify.com

When most people say “Spotify Model”, they’re only thinking Squads, Chapters, Tribes, Guilds



When it comes to product development culture, structure is the last thing you should be worried about, not the first.

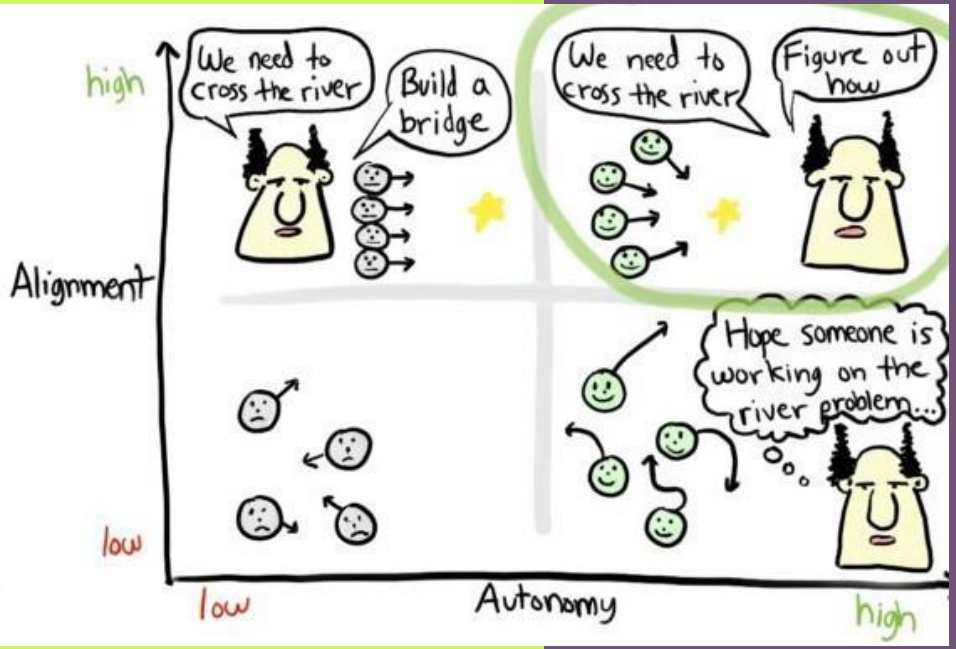
Without even looking at your context, the top 3 points that are more important:

01 Aligned autonomy

02 Trust-at-scale

03 Decoupling

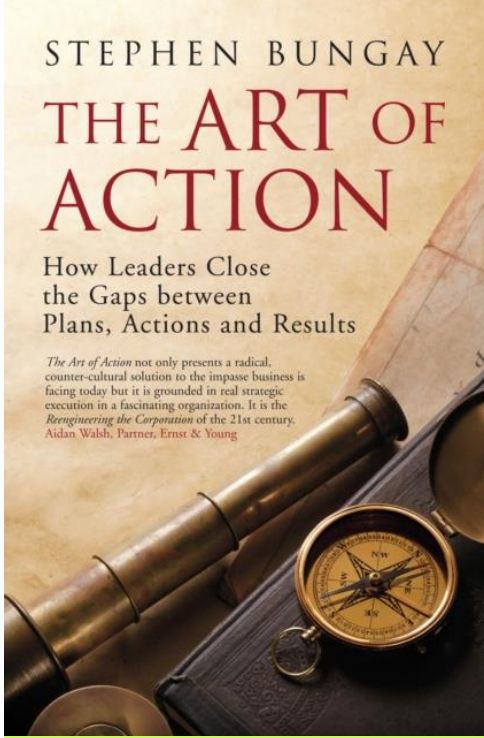
01.



Aligned autonomy

Figure 7 A choice?

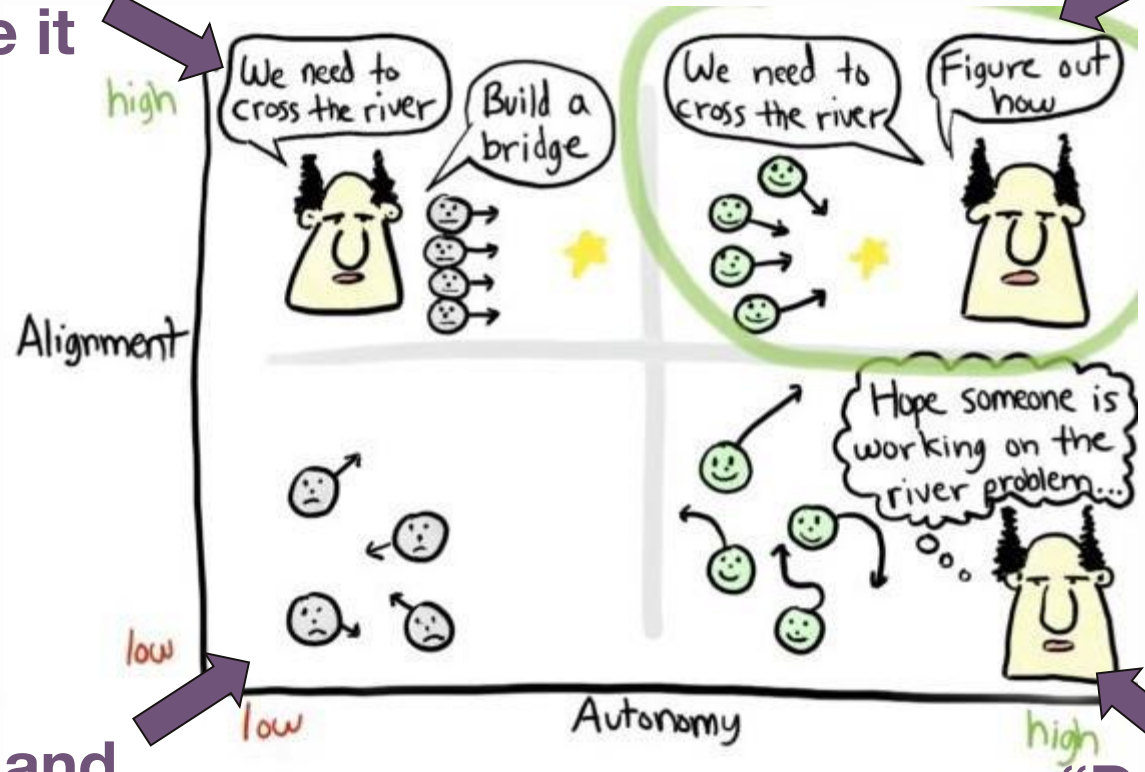
Von Moltke's insight is that there is no choice to make. Far from it, he demands *high* autonomy and *high* alignment *at one and the same time*. He breaks the compromise. He realizes quite simply that the more alignment you have, the more autonomy you can grant. The one enables the other. Instead of seeing of them as the end-points of a single line, he thinks about them as defining two dimensions, as in Figure 8.



Alignment and autonomy are not two ends on a scale but two dimensions on a 2x2 matrix.

Clear problem but told exactly how to solve it

Aligned autonomy



"Shut up and follow orders"

"Do whatever you feel like"

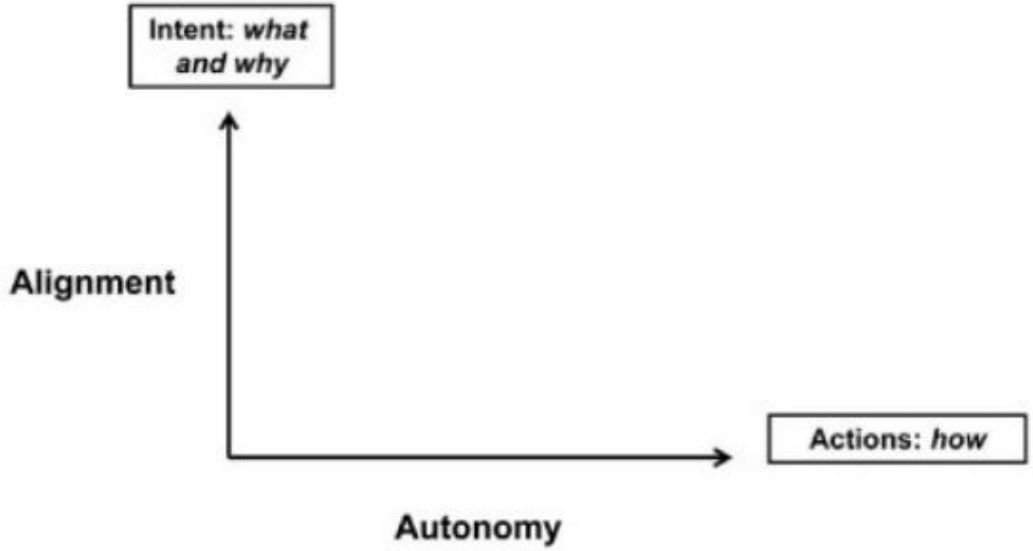
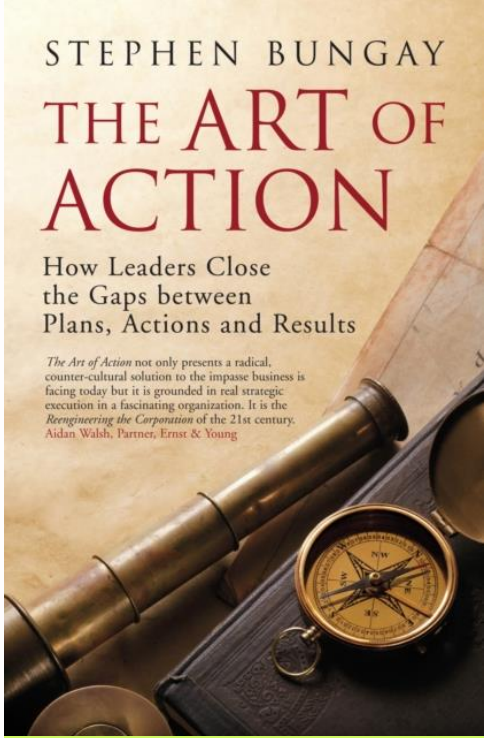


Figure 8 High alignment enables high autonomy

The insight is that alignment needs to be achieved around intent, and autonomy should be granted around actions. Intent is expressed in terms of *what to achieve and why*. Autonomy concerns the actions taken in order to realize the intent; in other words, about *what to do and how*. By requiring his



2 parts to aligned autonomy:

01

Clearly expressed product strategy

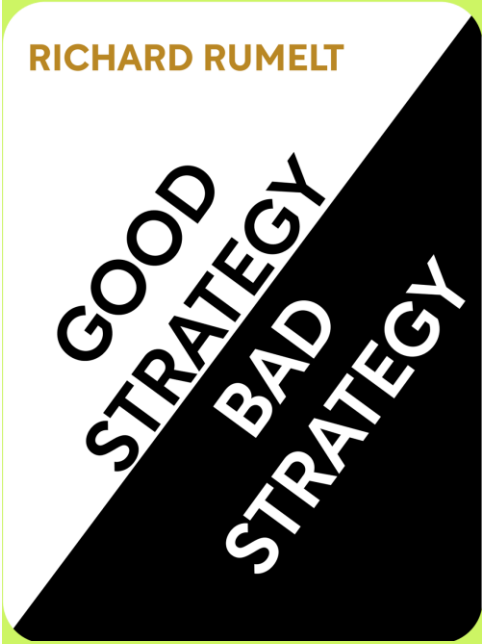
02

Empowered Product Teams (aka Squads)

Clearly expressed product strategy

The kernel of good product strategy:

1. Diagnosis based on data and insights;
2. Guiding policy, that is, a set of beliefs describing the general approach to overcome the identified problems;
3. A key set of coherent actions, or more accurately, bets



Example

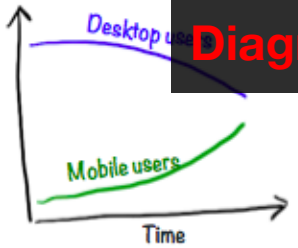
DIBB – an argument framework

Example:
(Something like this happened a few years ago)

Data ⇒ Insight ⇒ Belief ⇒ Bet

Diagnosis | **Guiding policy** | **Coherent action**

How people listen to music



Mobile is overtaking desktop as primary music gadget!

We have very few mobile devs compared to desktop

How we're staffed

WTF we're optimized for the wrong thing!

For long term survival, we need to become mobile-first

Hire a bunch of mobile devs

Train a bunch of our desktop devs into mobile devs

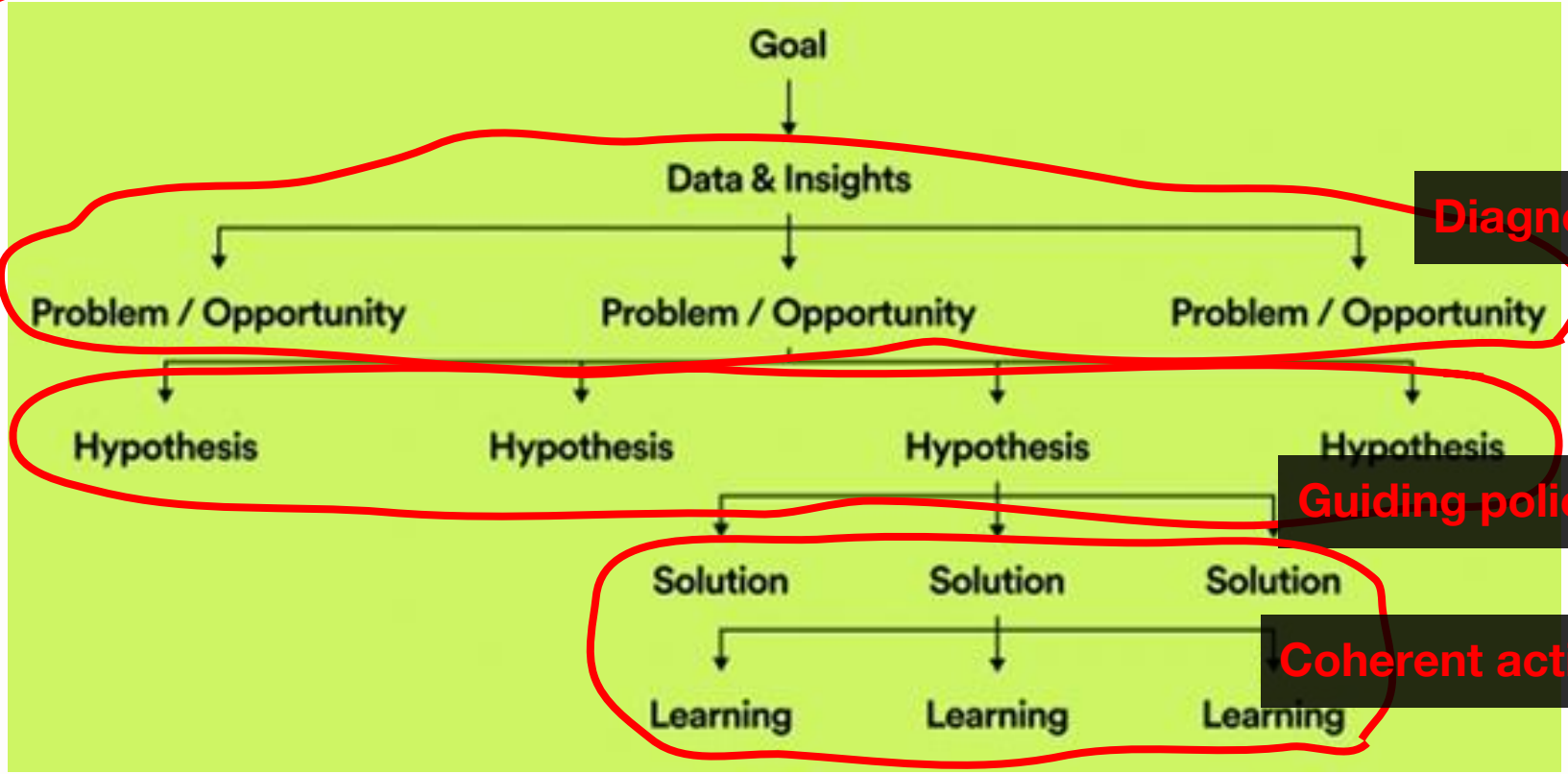
Build infrastructure for iterating fast on mobile

Feedback loop

https://blog.crisp.se/2016/06/08/henrikkniberg/spotify-rhythm

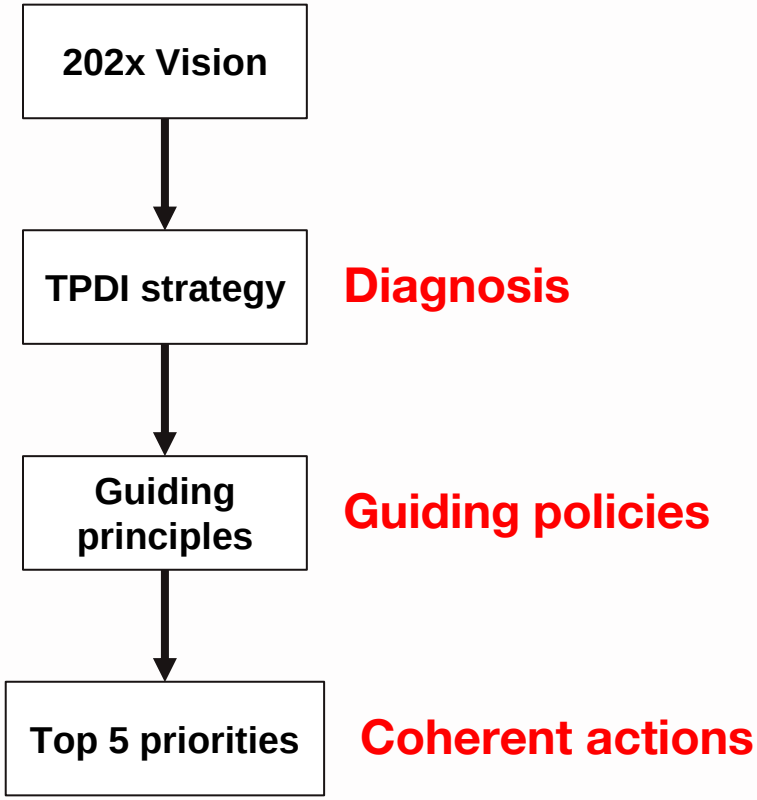
Example

Thoughtful Execution Framework



https://spotify.design/article/from-gut-to-plan-the-thoughtful-execution-framework

Example


202x Vision
TPDI strategy
Diagnosis
Guiding principles
Guiding policies
Top 5 priorities
Coherent actions

Empowered Product Teams (aka Squads)

“Squad” is not a synonym for “team”; it’s a synonym for what Marty Cagan calls an “Empowered Product Team”.

Squads aka “Empowered Product Teams”

Multi-disciplinary

Tech, Product, Design, Insights. All skills needed to progress the mission should exist on the Squad.

Have a mission

The individual Squad mission should fit within the context of the larger product strategy.

Expectation and authority to figure out how

Autonomy is not a benefit; it’s an expectation of responsibility. This includes coordinating with other Squads as-needed.

Source:

Example

Let's Talk Team Topologies

How to think about forming Squads in the newly proposed Amplify R&D org

Setting the Scene

- Newly formed Tribes will need to revisit Squad design
- Poorly designed Squad boundaries will lead to collaboration & coordination issues
- It is useful to provide principles & guidance for forming well-designed Squads

Foundational Beliefs

- It takes time for teams to reach the "performing" stage, including forming agreed ways of working & broadly shared values
- Autonomous teams are more motivated & responsive; autonomous teams are aligned through clarity of shared goals
- There are human cognitive limits to social relationships, and therefore trust
- Reporting structure != value creation structure; optimize for the overall flow of value creation, not any one area or team
- Teams should be the elemental org design building block, not individuals
- Collaboration is expensive, so it should be intentional
- Ownership of systems / services should not be shared across teams
- Team responsibilities should be limited to manage cognitive load, i.e. 1 complex domain OR 1 complicated domain with 2-3 simple domains
- Conway's Law -- architecture is constrained by org structure; but we can utilize org design to encourage the right architecture

Four Types of Teams

Stream Aligned Teams
- *Aligned to a single stream of work, i.e. single product, service, set of features, user journey, persona, etc.*
- *Should not require hand-offs to perform work.*

Platform Teams
- *Enable stream aligned teams to deliver autonomously*

Complicated Subsystem Teams
- *Build and maintain subsystems that require deep specialist knowledge*

Enabling Teams
- *Technical consulting teams, typically specialists who can bridge capability gaps, i.e. research, trying different options, suggest tooling / practices / frameworks, etc.*

Three Inter-Team Interaction Modes

Collaboration
- *Two teams work closely together for a defined period of time*
- *Synchronous communication works better*
- *Can be utilized to discover X-as-a-Service interactions*

X-as-a-Service
- *One team consumes a service / API provided by another team*
- *Enables asynchronous communication opportunities*

Facilitating
- *One team helps another team learn or adopt a new approach / technology for a defined period of time*

Team Types & Standard Interaction Modes

	Collaboration	X-as-a-Service	Facilitating
Stream-aligned	Typical	Typical	Occasional
Enabling	Occasional		Typical
Complicated-subsystem	Occasional	Typical	
Platform	Occasional	Typical	

High-Level Sequencing

Step 1
Identify domains / value streams

→

Step 2
Identify Stream Aligned teams

→

Step 3
Identify Platform, Complicated Subsystem, and Enabling teams as required

Mural by: Jason Yip & Sharon Malfesi

Credit: "Team Topologies: Organized Business and Technology Teams for Fast Flow" by Matthew Skelton & Manual Pais

02.

Trust-at-scale

2 contributors to trust-at-scale:

01

Cross-pollination

02

Culture of mutual respect (aka People > *)

Cross-pollination

Cross-pollination humanises across boundaries

Embedding

Someone temporarily transfers to another team.

Liaisons

Someone acts as the primary point-of-contact to another team. Effective liaisons are deliberately selected based on being well-respected and reliably good at developing relationships.

Internal movement

Someone permanently transfers to another team but still has relationships with their previous team

Source:

Example

(Cara Lemon) Squad interaction modes

- Consult to not own;
- Consult to own;
- Temporary embed (outgoing);
- Temporary embed (incoming);
- Form temporary team;
- Build to hand off;
- Build to own

https://jchyip.medium.com/team-interaction-modes-5e550789e41b

Culture of mutual respect (aka People > *)

Mutual respect encourages trust

Role models

Influential people consistently model respectful behaviour.

Systems

Systems incentivise respectful behaviour and dis-incentivise disrespectful behaviour.

Stories

Stories about what is good emphasise respect; stories about what is bad emphasise disrespect.

Source:

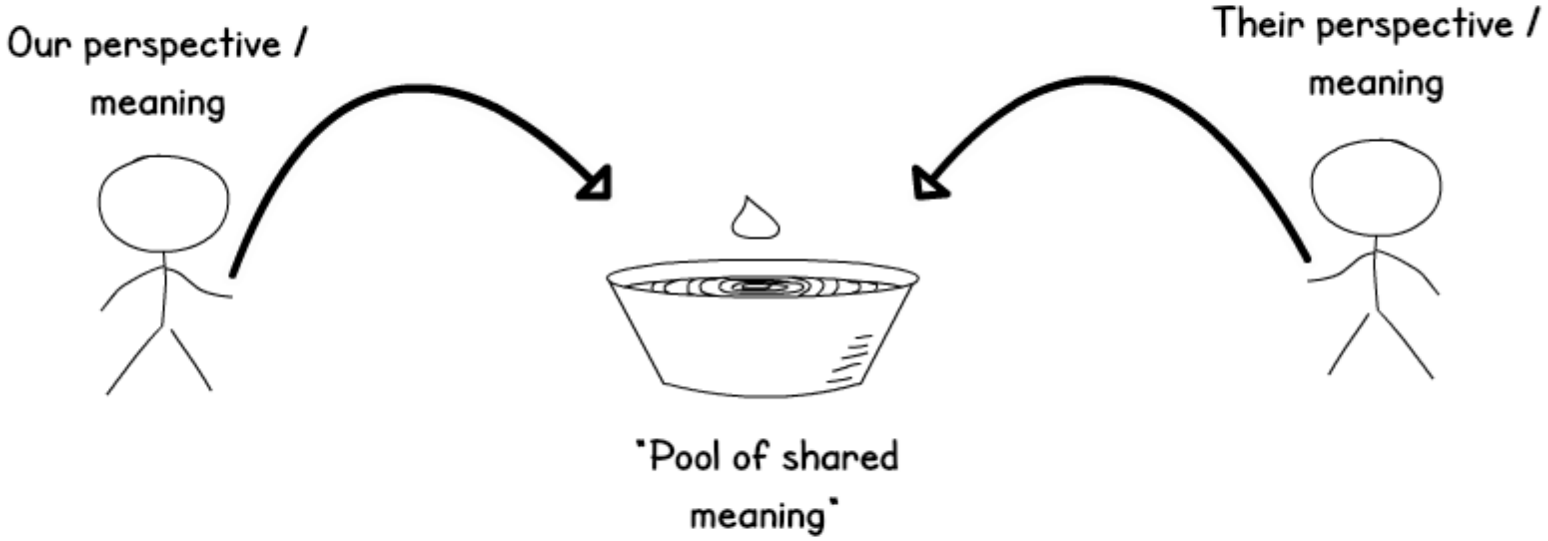
“Anything that is human is mentionable, and anything that is mentionable is manageable.”

Fred Rogers

No to pretending politics and fear don’t happen; yes to encouraging dialogue and safety in response to politics and fear.

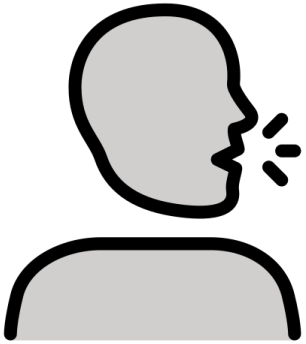
Example

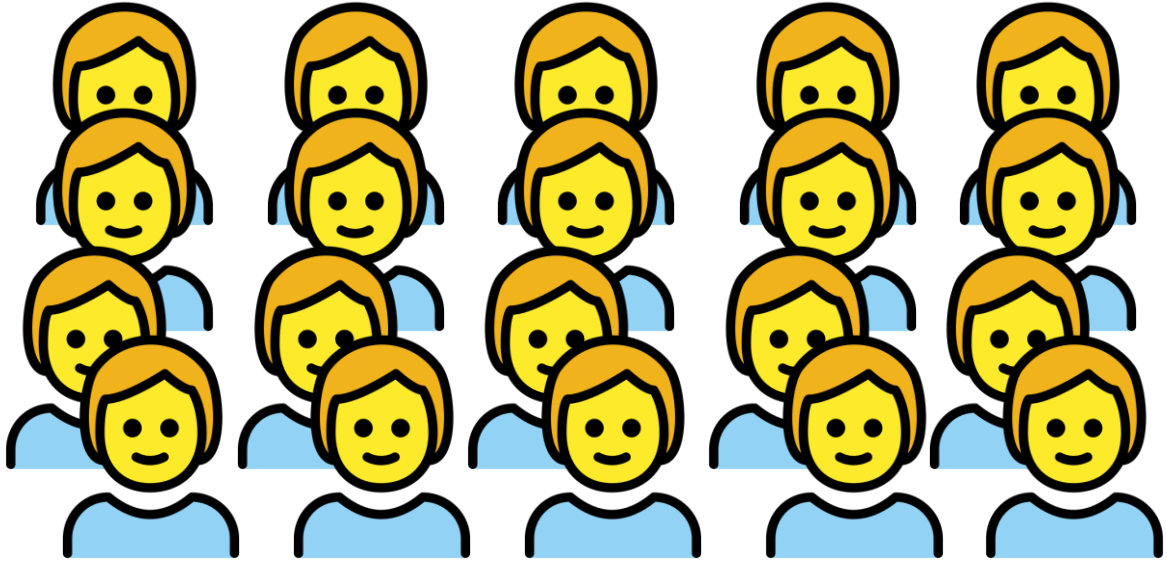
Mutual respect means committing to shared meaning



Concept from Crucial Conversations

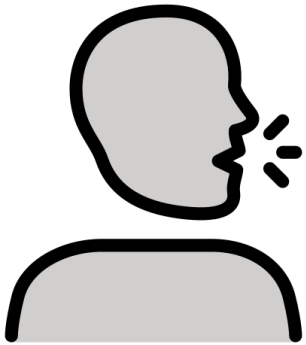
Example

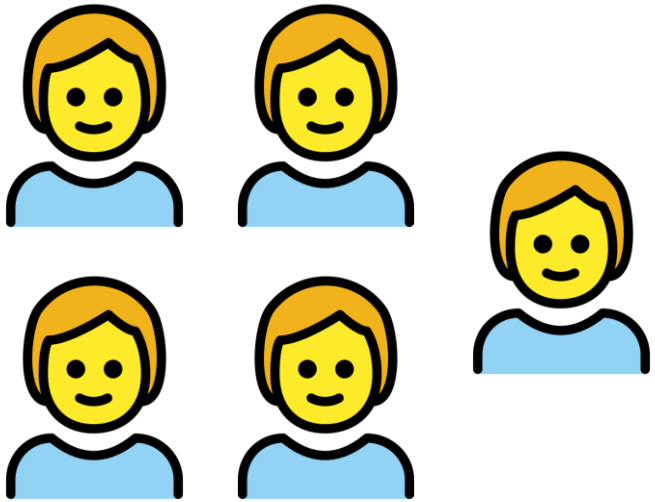




This doesn’t get any back-and-forth.

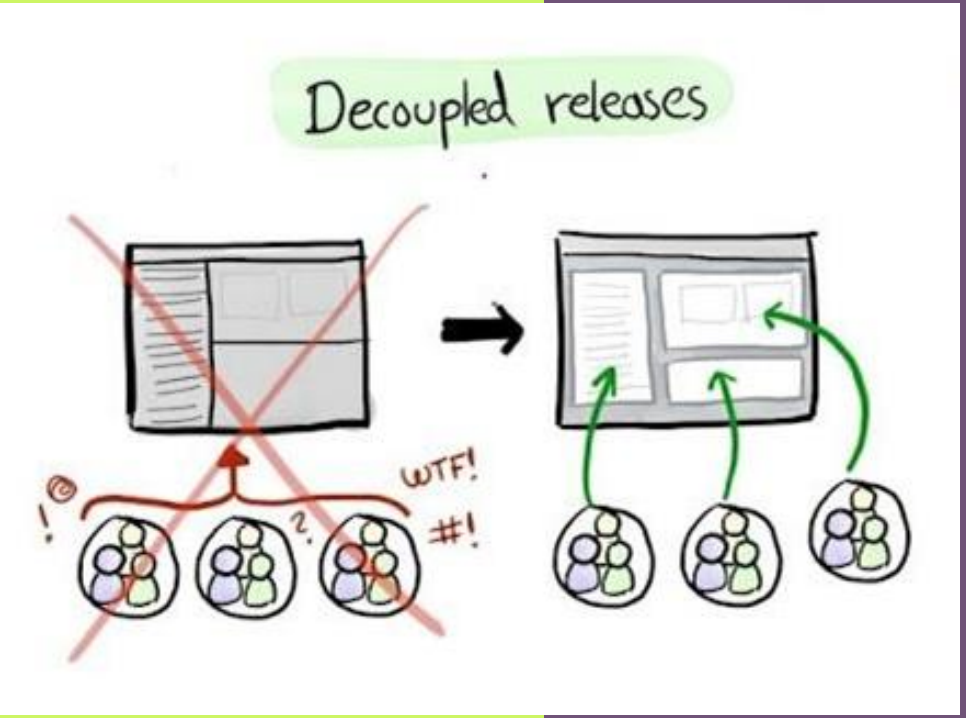
Example





This does (and shows respect)

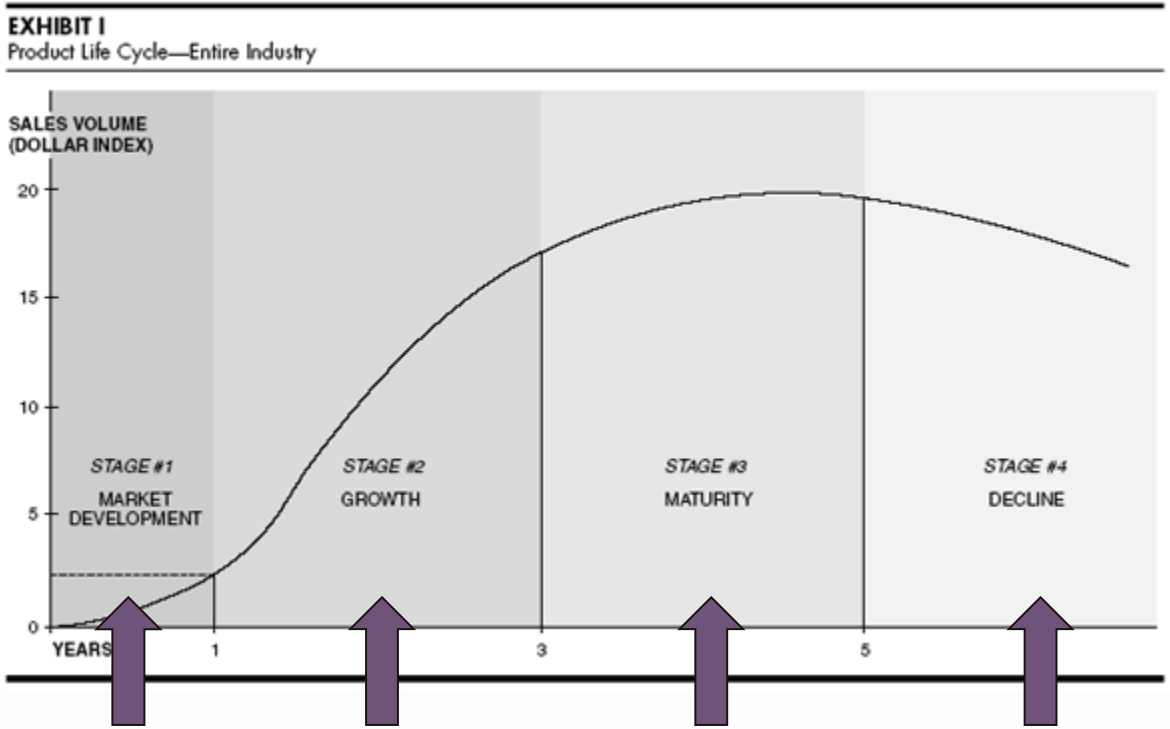
03.



Decoupling

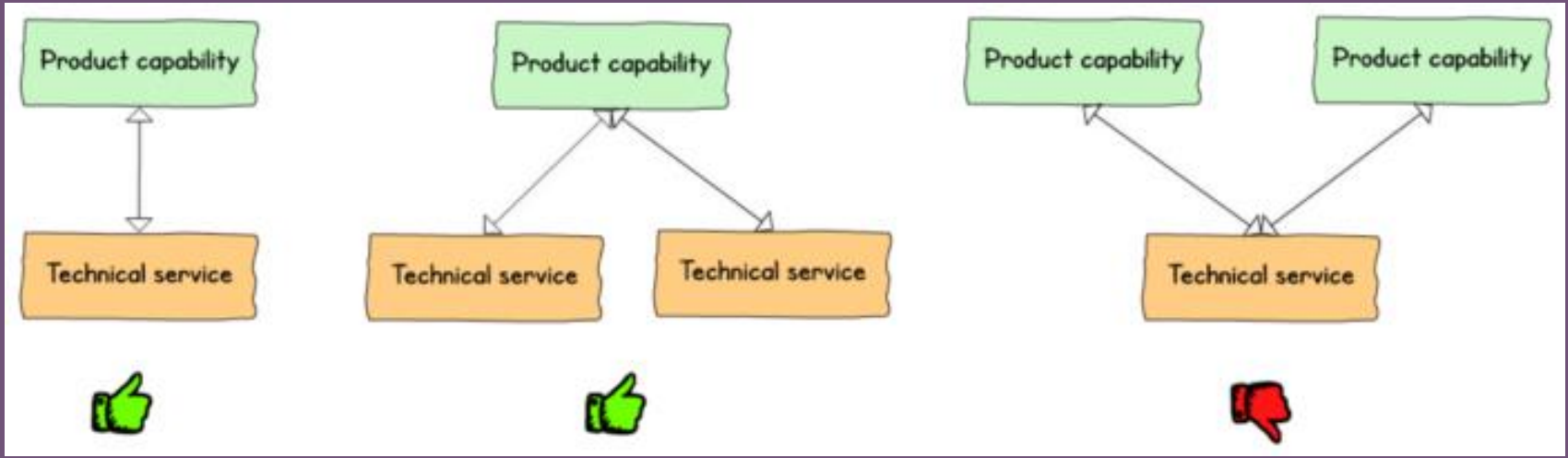
Architecture should be coupled where product capabilities should be coupled and decoupled where product capabilities should be decoupled.

"Exploit the Product Life Cycle" by Theodore Leavitt, Harvard Business Review Nov 1965



Experimentation **Iteration** **Optimisation** **Outsourcing / shutdown**

Technical services supporting product capabilities should not cause capabilities at different stages to be coupled.



Services supporting stable capabilities should not depend on services supporting more volatile capabilities.

Example

	Commodity / hygiene		Differentiating / better = $		Innovation / Not guaranteed to work
Product capabilities	Account management	Payments	Advanced forecasting	Advanced pricing	New formats
	Audience management	Campaign management	Decision optimisation	New client platforms	New targeting
	Commodity 3rd party measurement		Reporting		
			Customer-specific measurement?		
Architecture services	Ad serving console	Mobile client platform	Ad Studio	Native layer features	?
	Order management	Audiences	Partner API	Format business logic	
	Budget allocator	Google Ad Manager integration	Spotify for Artists integration		

The top 3 points that are more important than Squads, Chapters, Tribes, Guilds:

01 Aligned autonomy

02 Trust-at-scale

03 Decoupling

THANK YOU

Spotify