Real-life Agile Scaling

Keynote - Agile Tour Bangkok
Nov 21, 2015

Henrik Kniberg

henrik.kniberg@crisp.se
@HenrikKniberg

Consultant

crisp
www.crisp.se

Parent

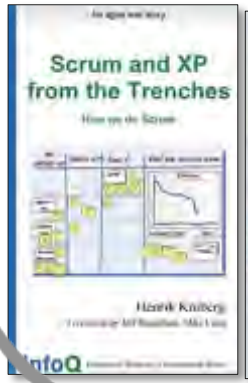
Agile & Lean coach

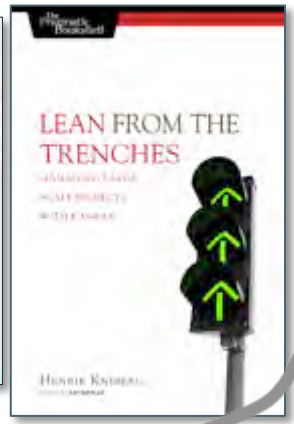
Spotify

LEGO

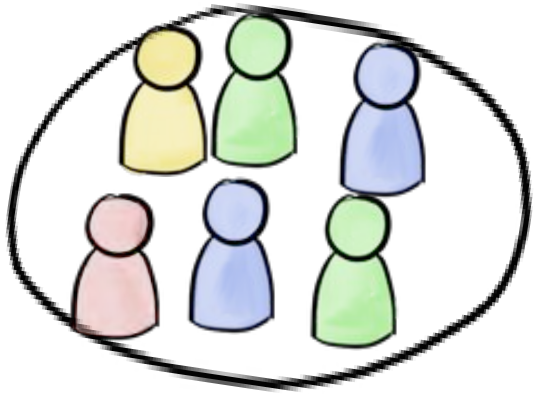
skype

Author

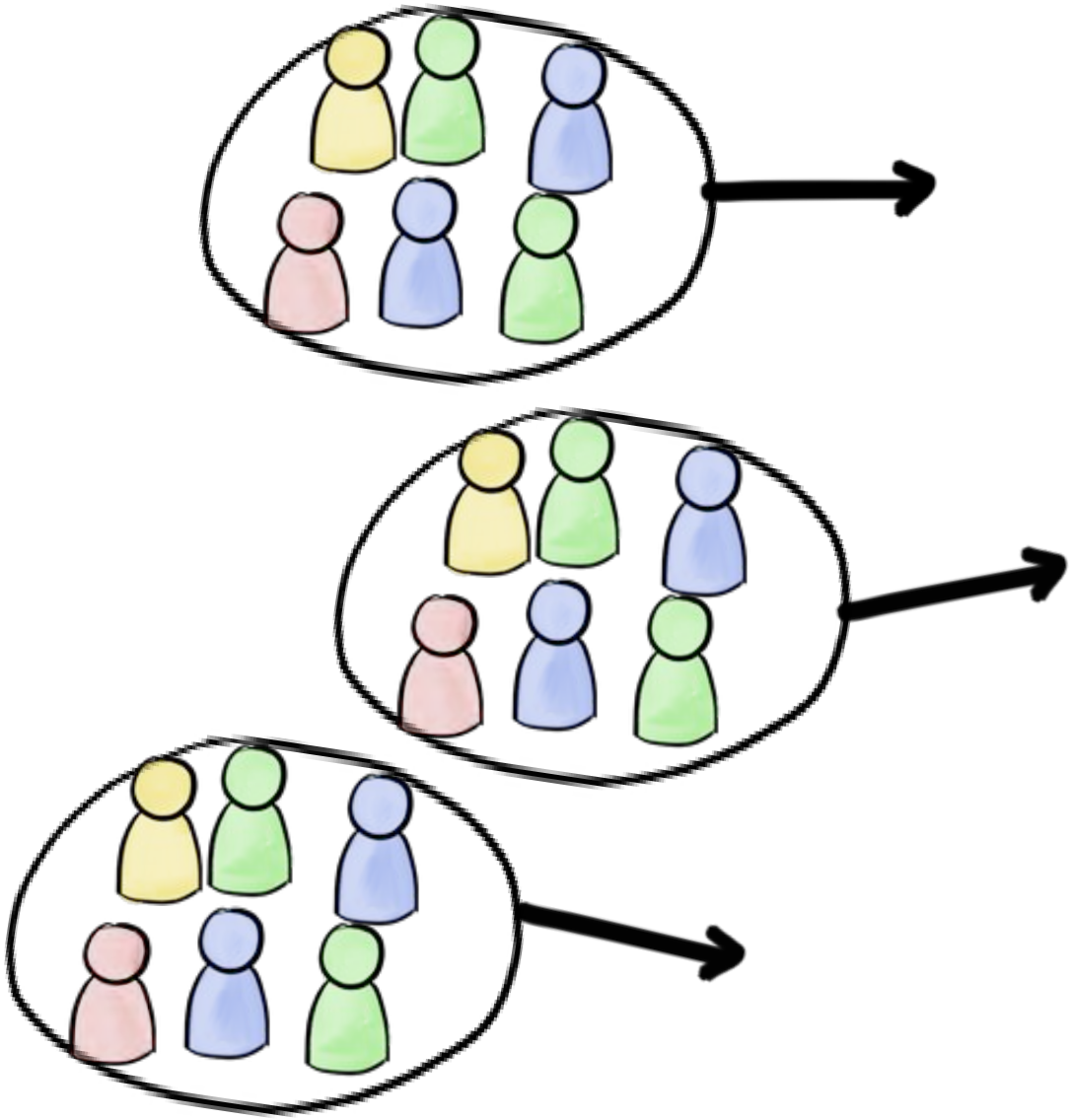




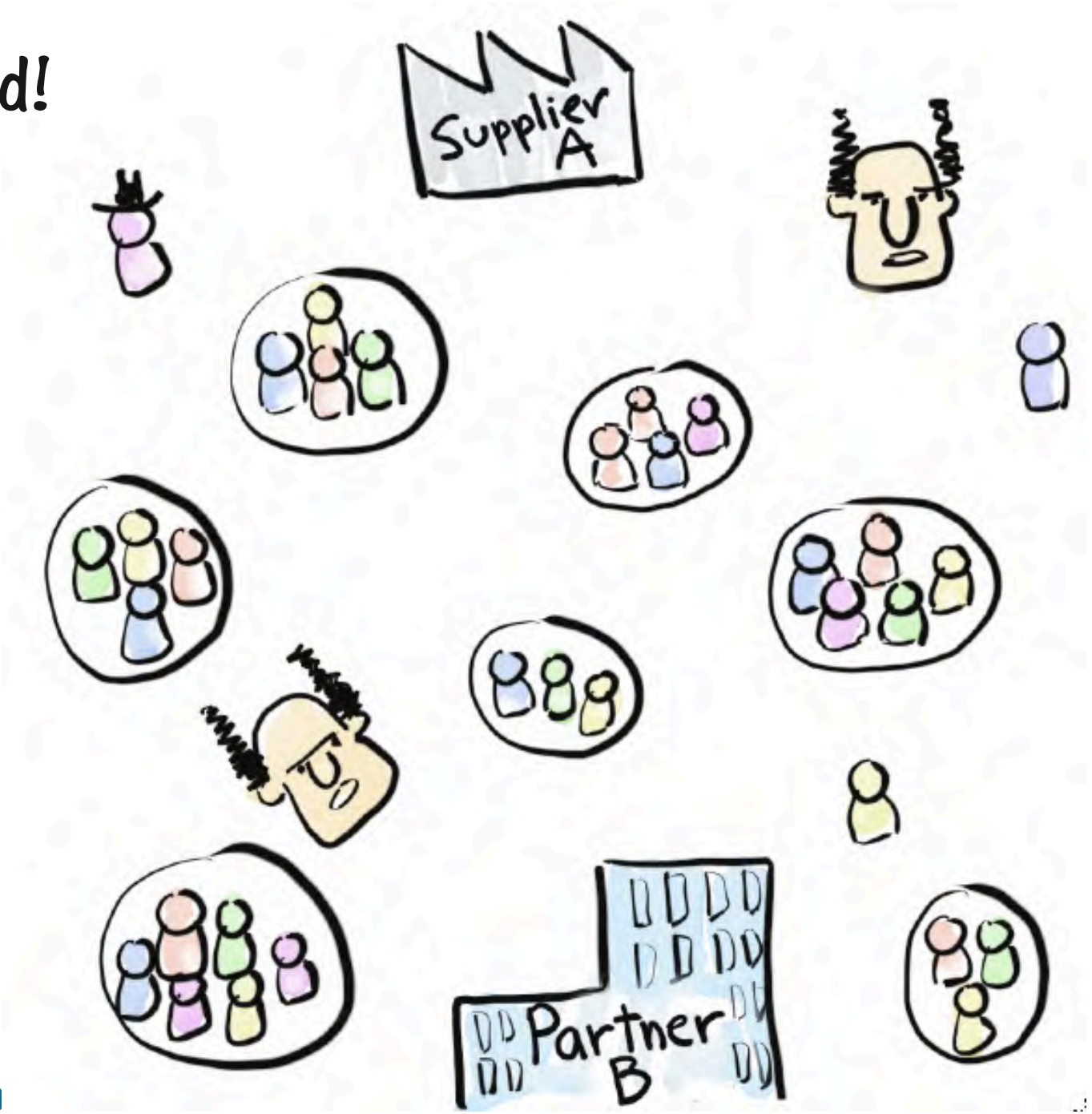
Not too hard



A bit trickier

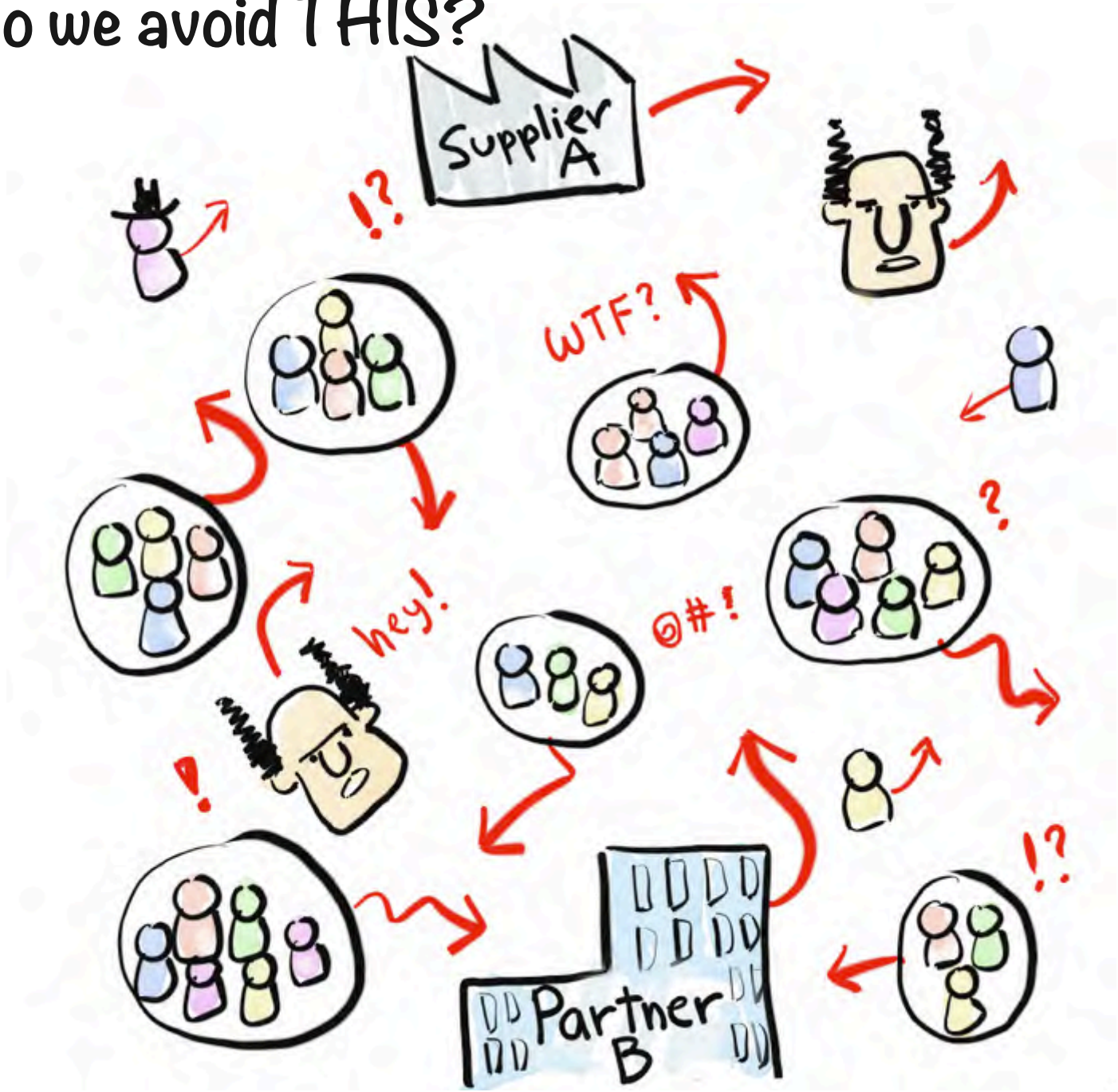


Hard!



Henrik Kniberg

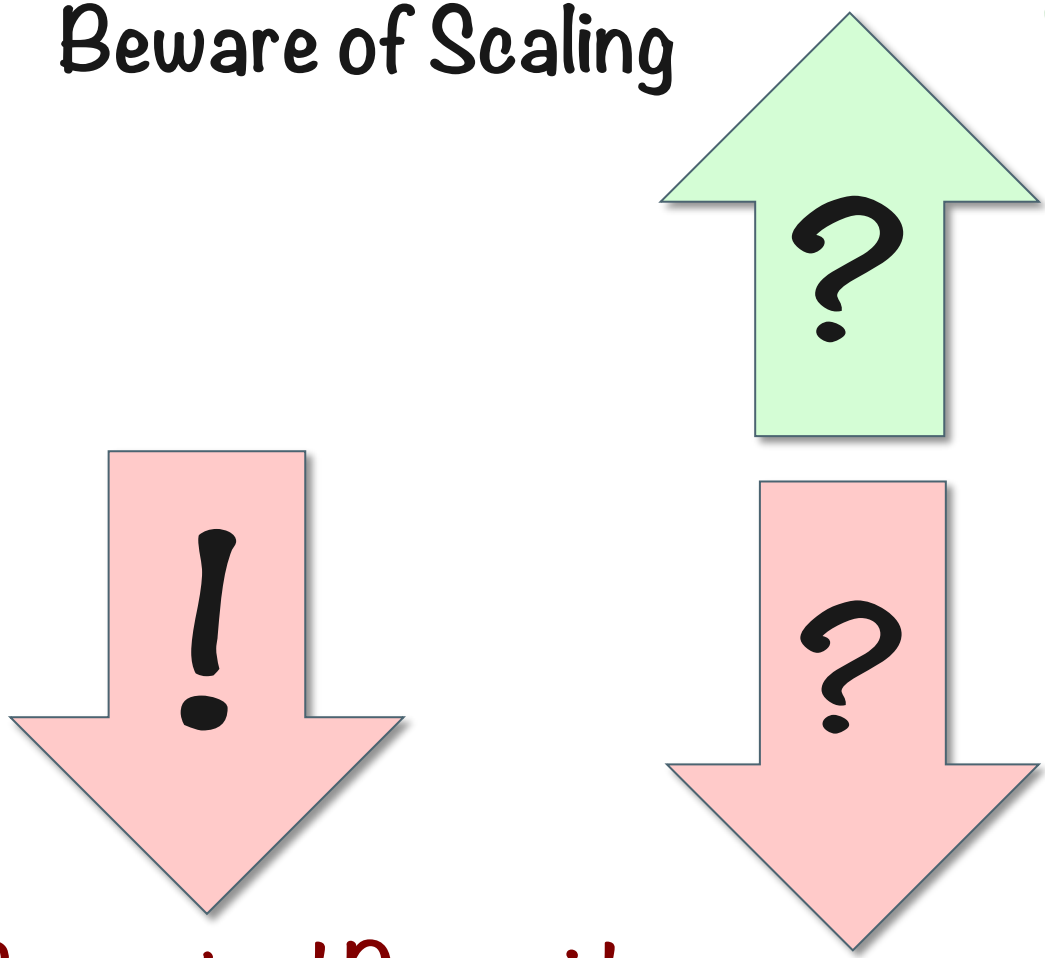
How do we avoid THIS?



Henrik Kniberg

Beware of Scaling

Beware of Scaling



Potential Upside:

- **Shorter delivery time**
  more hands on deck,
  parallell work
- **Better product**
  access to wider range
  of competences

Guaranteed Downside

- Cost
- Complexity

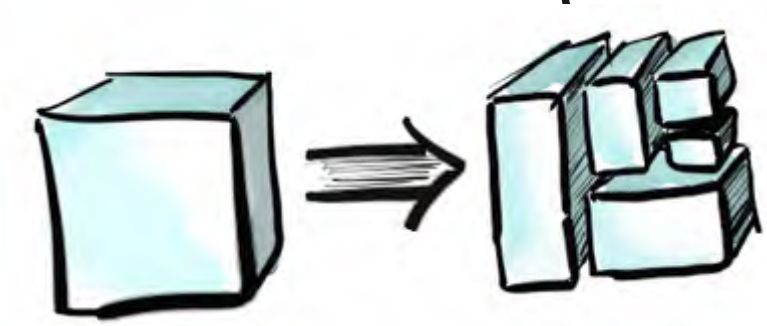
Potential Downside (Risk):

- **Longer delivery time**
  because of misalignment
  & dependencies
- **Worse product**
  because of
  bad communication

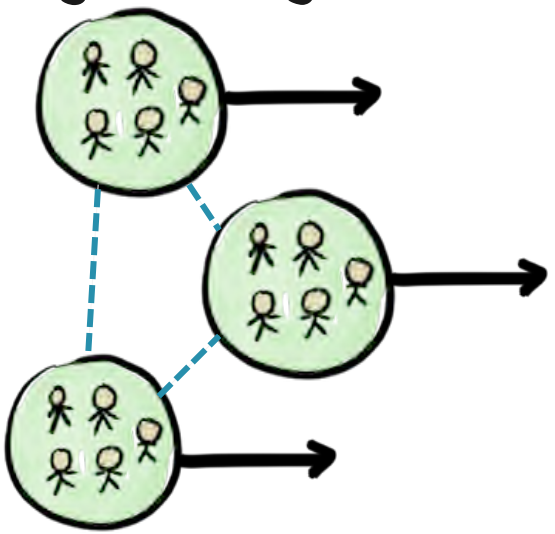
Henrik Kniberg

Stuff you need to figure out with multiple teams

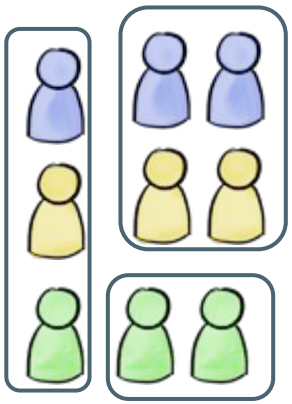
How to slice the elephant



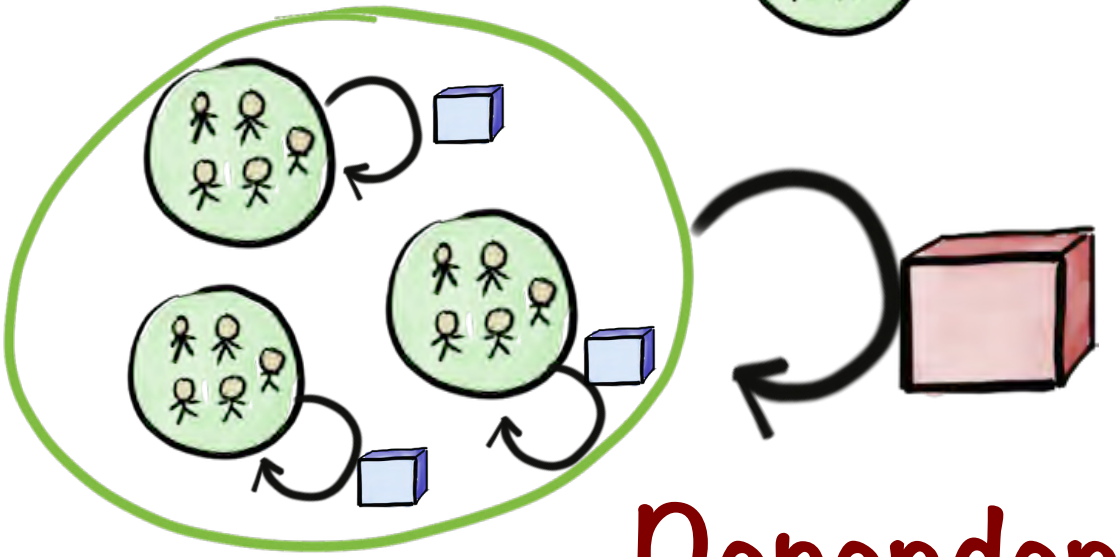
Team sync / alignment



Team structure



Feedback loop

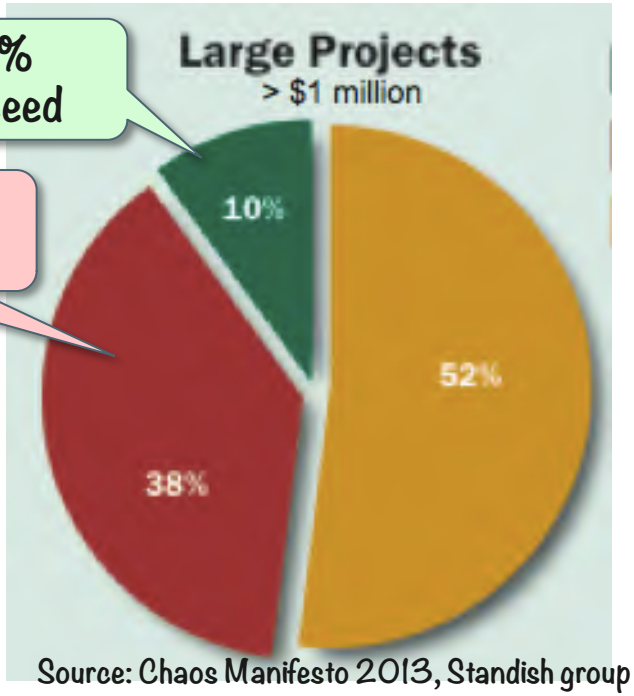


Dependencies!

Henrik Kniberg

Slice the elephant

Big Bang = Big Risk



Source: Chaos Manifesto 2013, Standish group

Henrik Kniberg

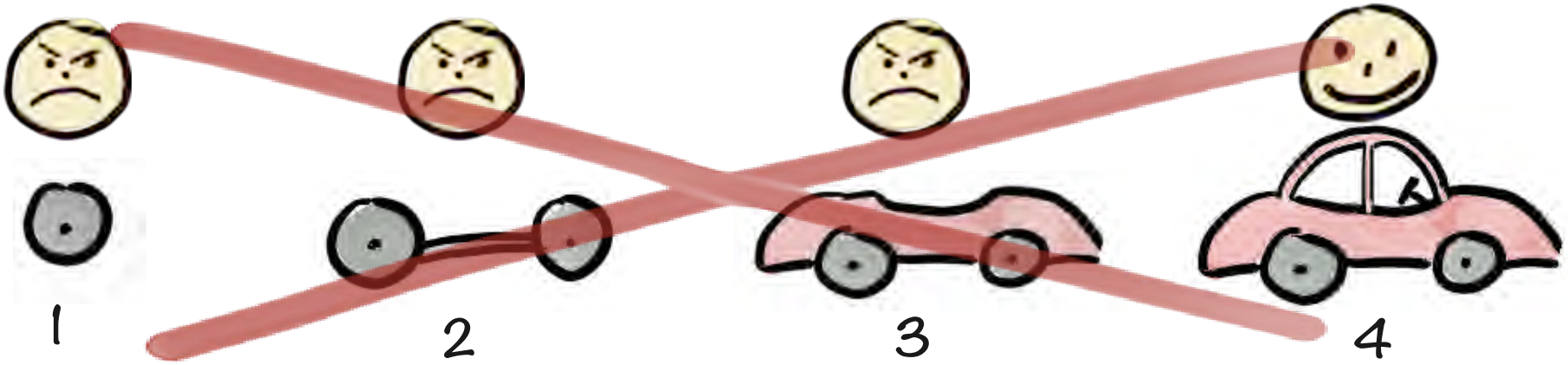
≈250 people involved

4 years to first public release

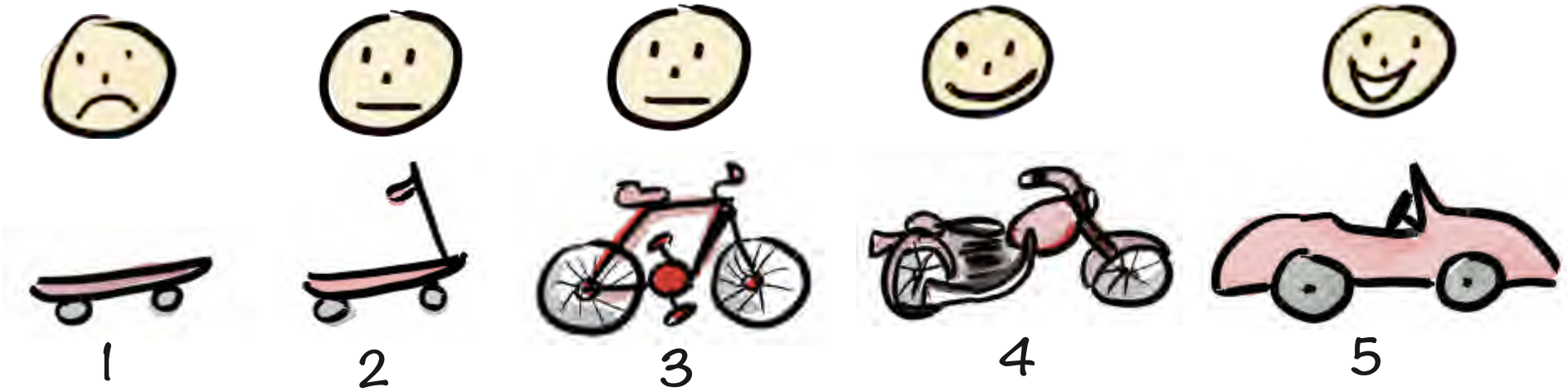
Shut down after 2 years of operation

Henrik Kniberg

Not like this....



Like this!

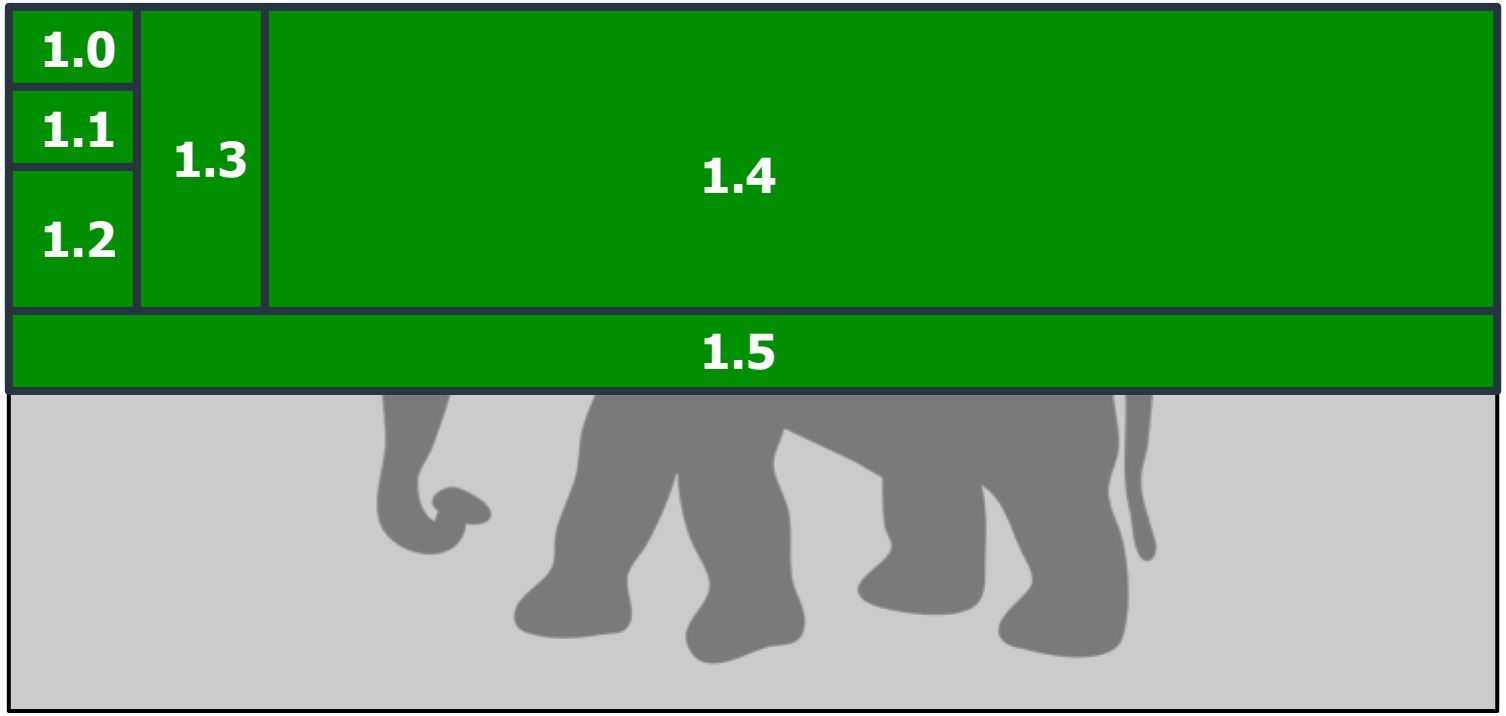


Henrik Kniberg

Slice the elephant!

Region
Östergötland,
Uppsala, etc

Crime types
(weapon,
drunk driving,
shoplifting, etc)



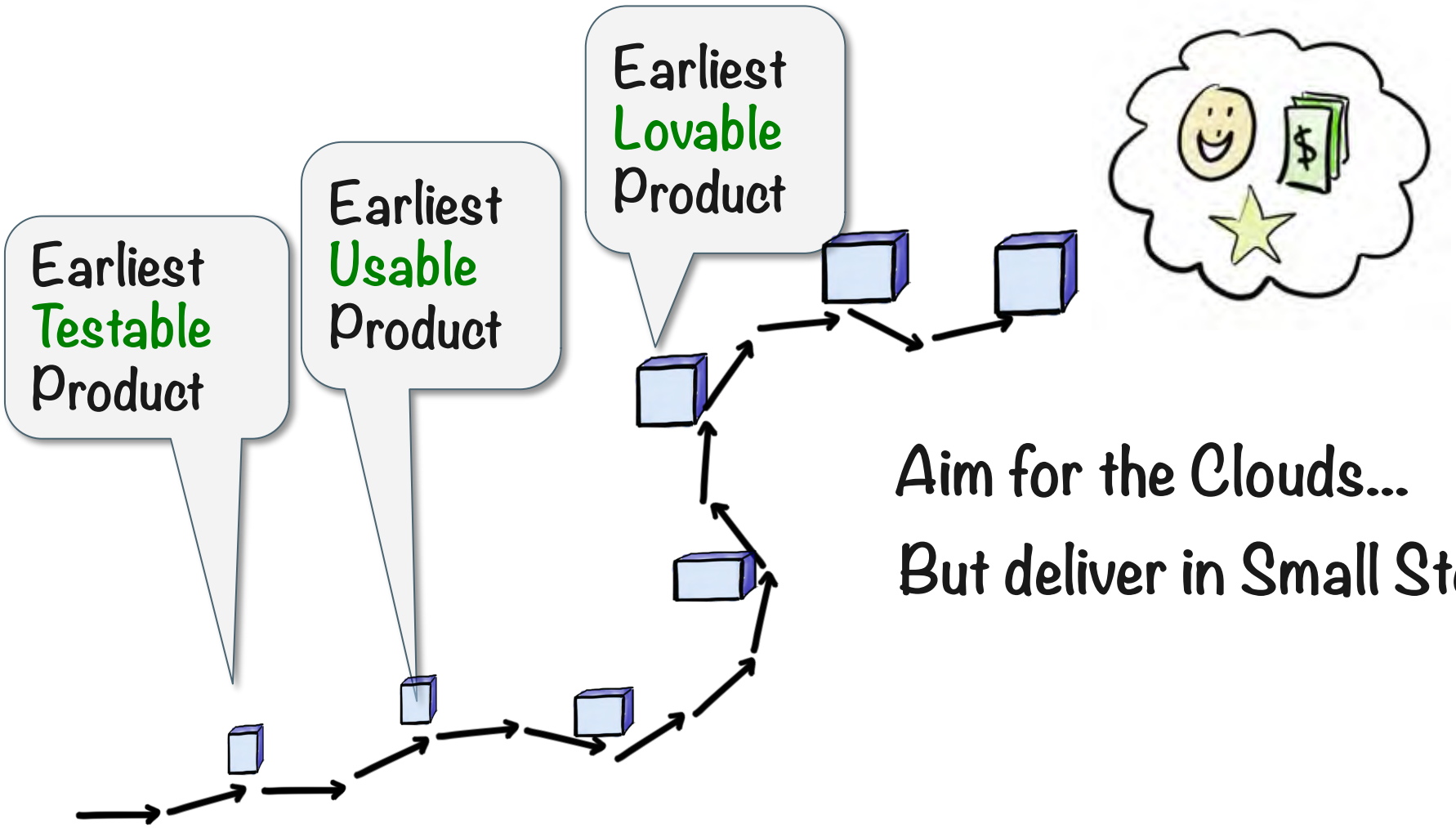
Integrations

Henrik Kniberg

MVP – Minimum Viable Product

Henrik Kniberg

Minimum viable ⇒ Earliest testable/usable/lovable

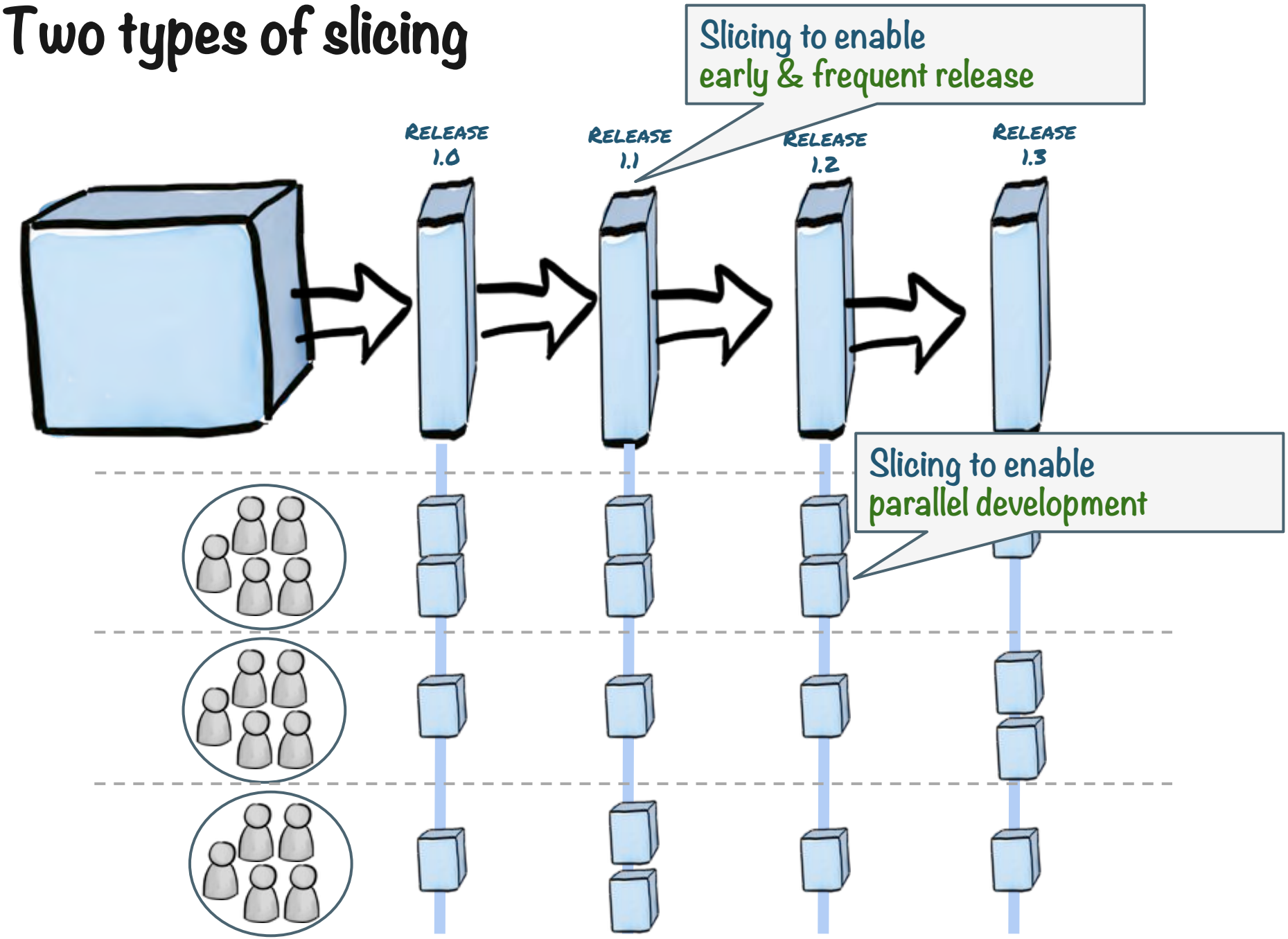


Aim for the Clouds...

But deliver in Small Steps

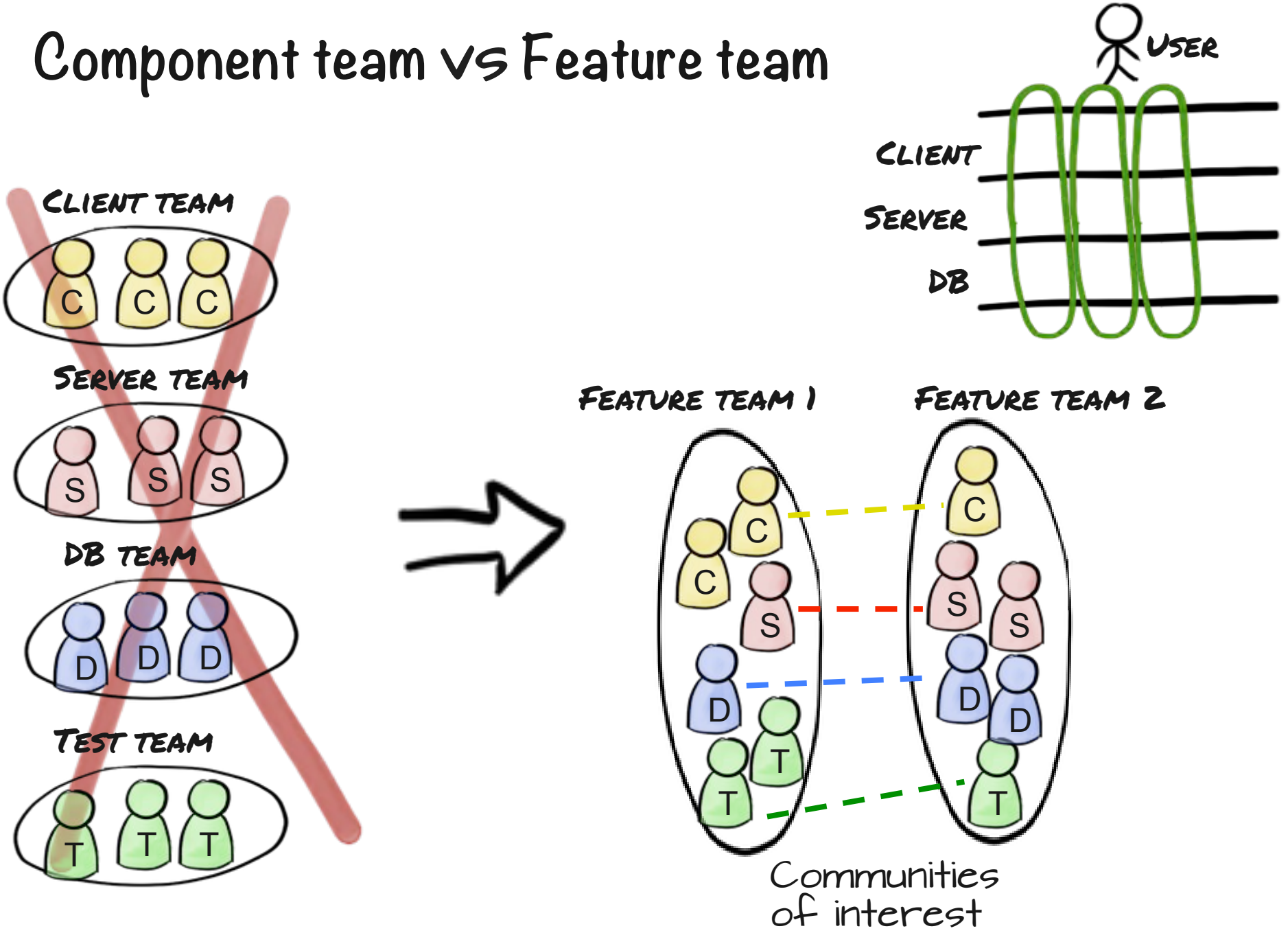
Henrik Kniberg

Two types of slicing



Build a suitable Team Structure

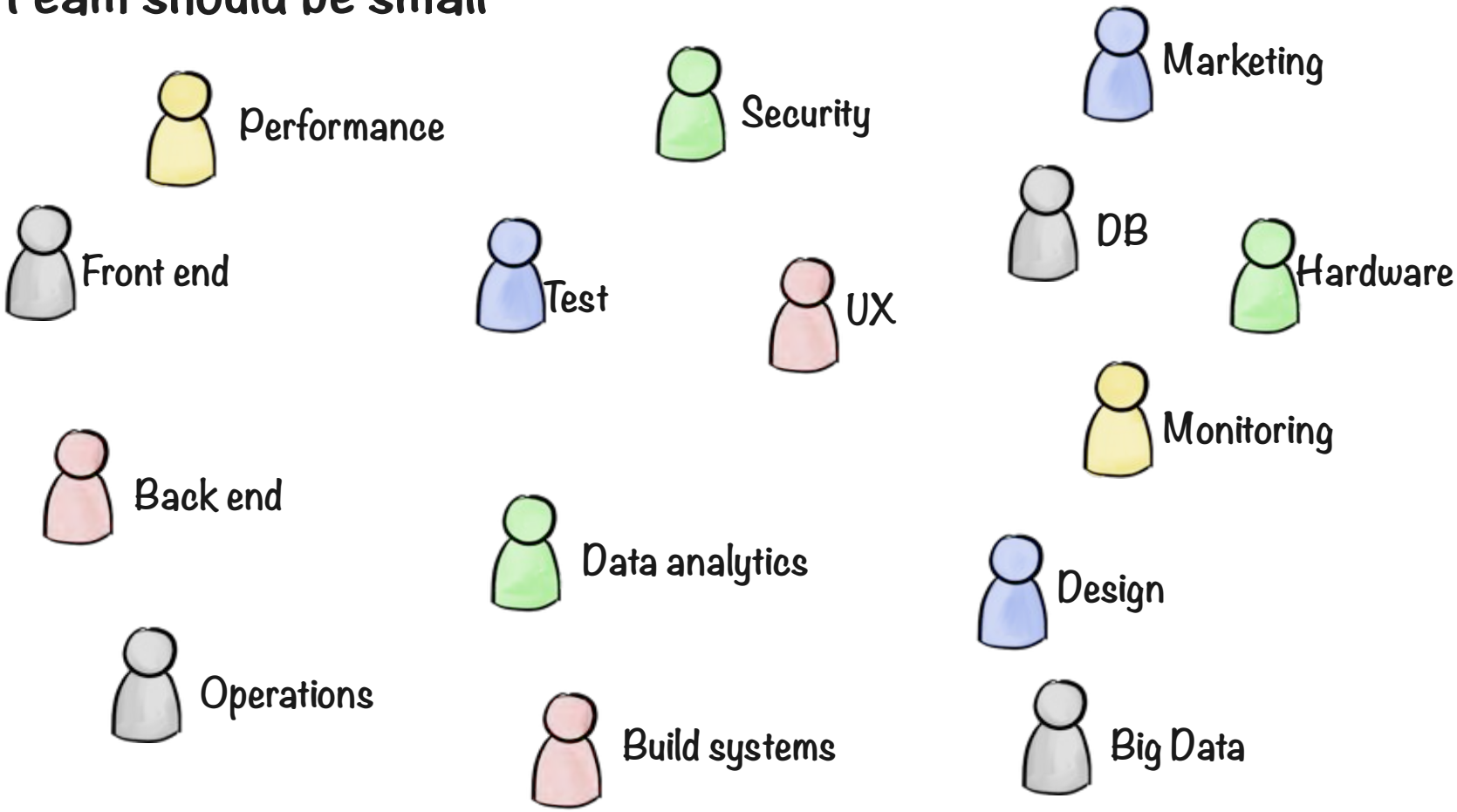
Component team vs Feature team



Henrik Kniberg

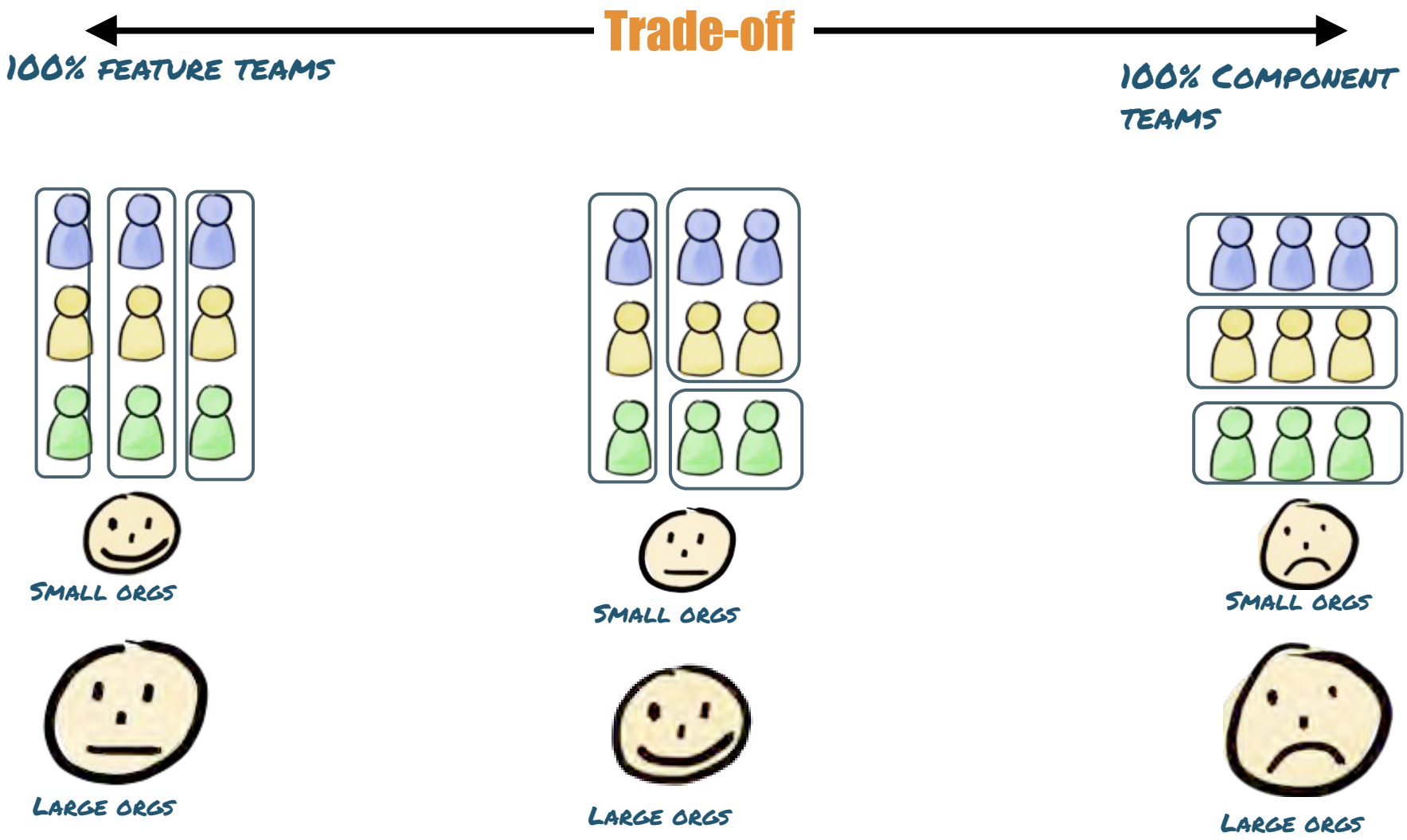
Two conflicting goals (at scale):

1. Team should be "full-stack"
2. Team should be small



Henrik Kniberg

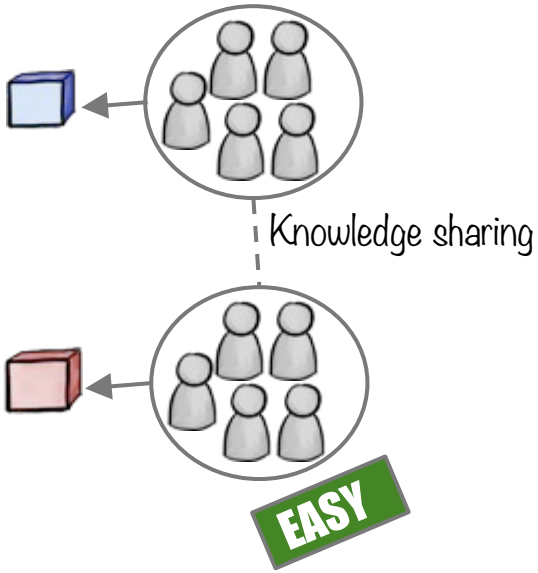
Team types - finding the right balance



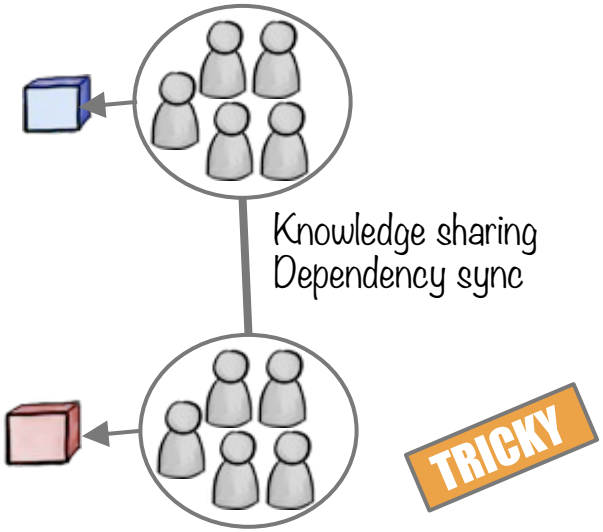
Henrik Kniberg

Types of dependencies

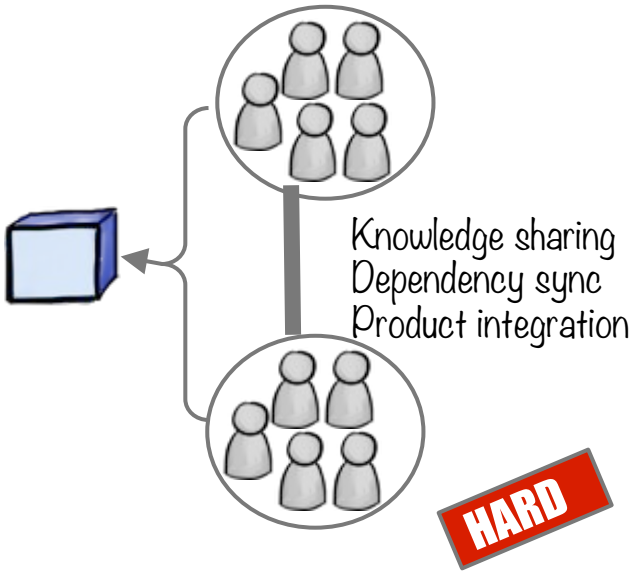
independent teams



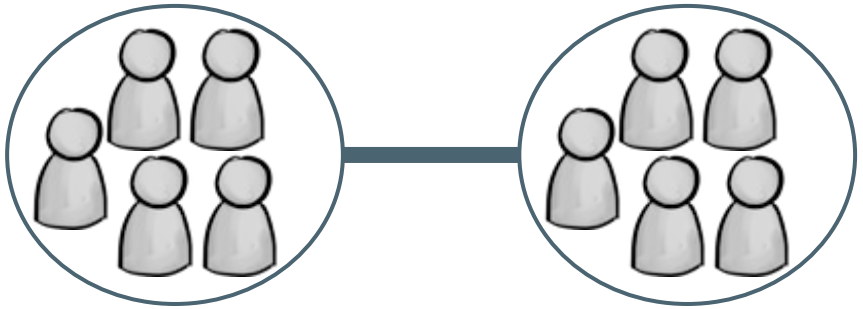
Building different products, but have dependencies



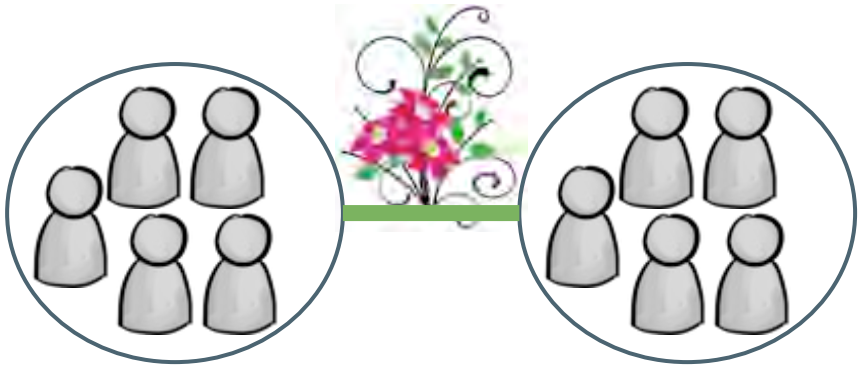
Building the same product (implicit dependency!)



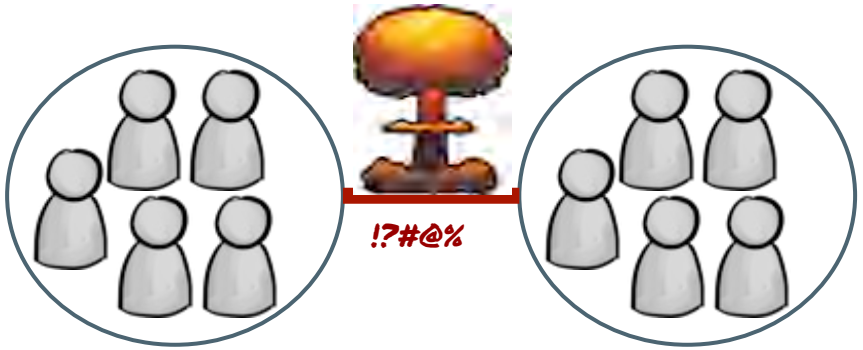
Dependencies



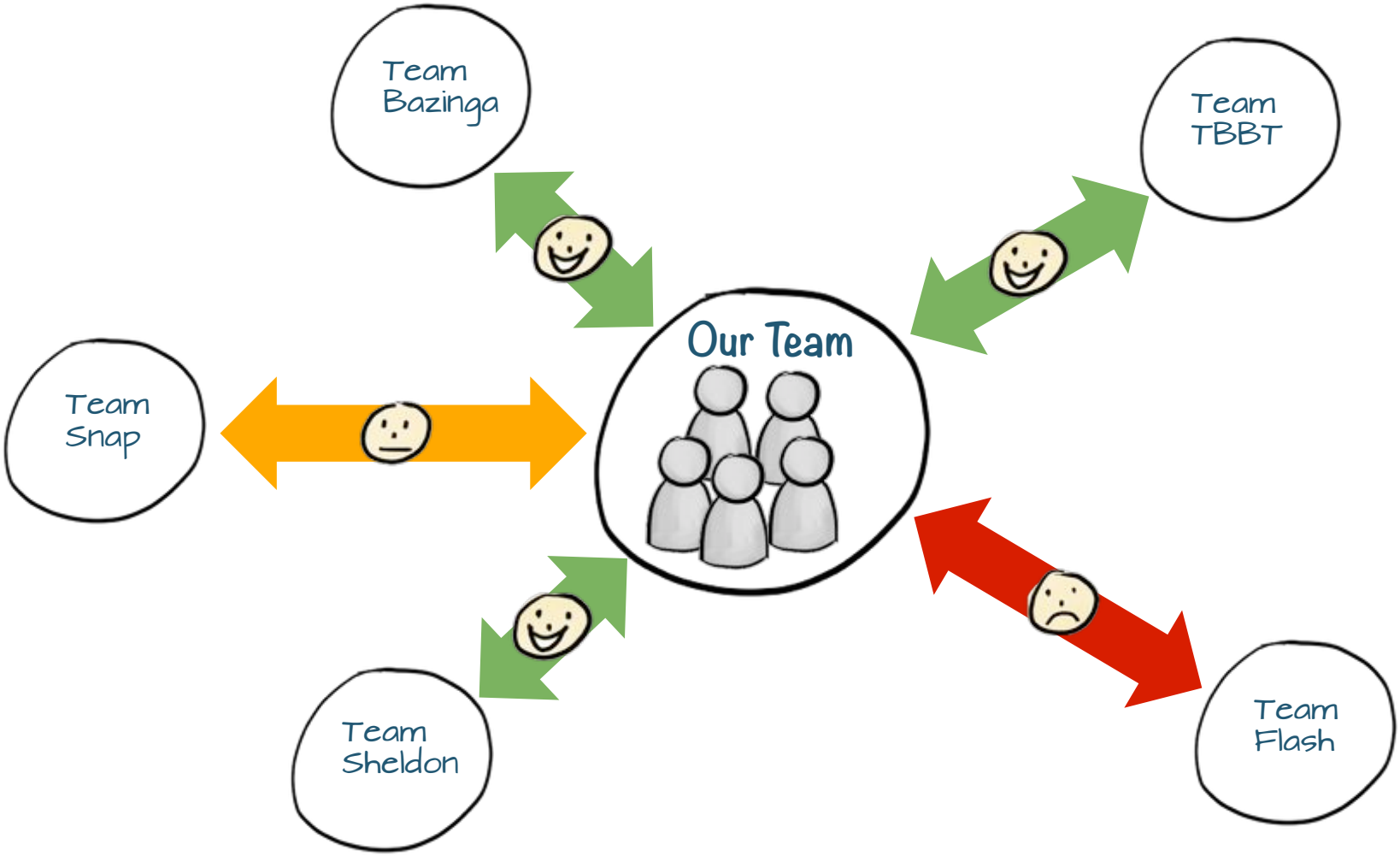
Good Dependency
(aka "collaboration")



Bad Dependency



Example:
Visualizing team dependencies



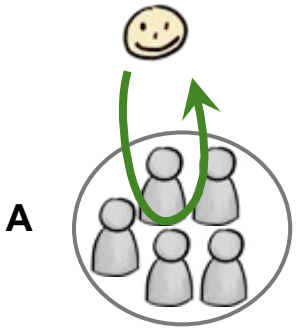
Henrik Kniberg & Jan Grape

Example:
Visualizing team dependencies

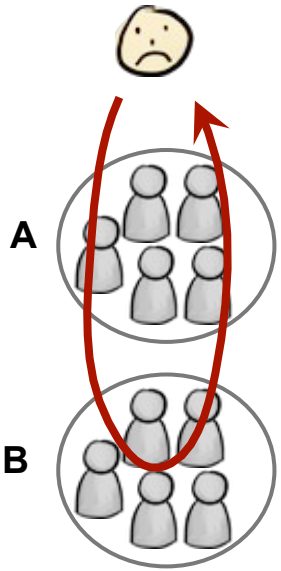
	A	B	C	D	E
1	**Squad**	**Depends on**	**Dependency**	**Comment**	**Same tribe?**
2	**Music Player**				
3	Content	Ops	Slowing	Need machines, connections, help set-up things etc. Works really well in general, but at times the workload on operations causes the lead times to grow and slow us down	No
4	Content	NeXT	No problem	Storage. Not big, mostly information/communication needs to happen	No
5	Content	BFS	No problem	Replacement service	Yes
6	Content	Team 2	No problem	Communication around next story	No
7	Content	Team 1	Future	Content ingestion	No
8	BFS	UX	Slowing	Need UX to discuss, review and provide mock-ups	No
9	BFS	Content	No problem	Normal dependencies, sprint work	Yes
10	BFS	Mobile	Slowing	No internal mobile developers within Squad	No
11	BFS	Analytics	Slowing	A/B test results slowing down roll outs of features	No
12	BFS	Team 3	Slowing	Waiting for data dumps	No
13	BFS	Team 1	Future	Waiting for data dumps	No

Good vs bad dependencies

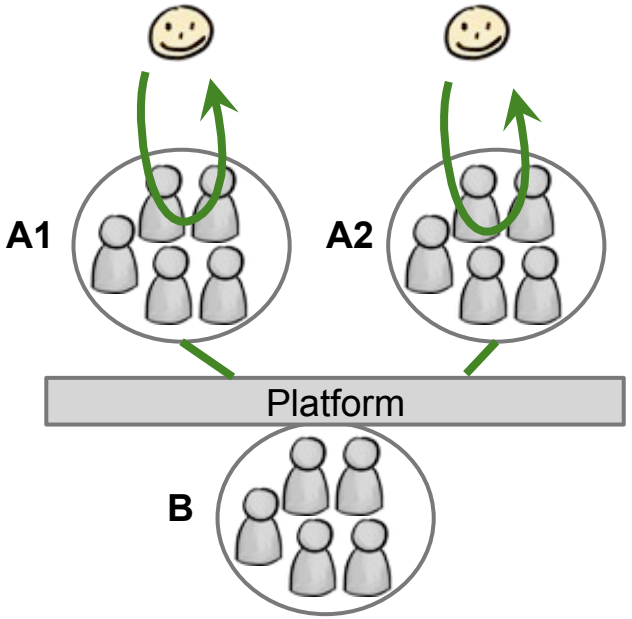
Full-stack team.
Can deliver customer value independently.



Coupled teams
A must sync with B in order to deliver customer value.



Platformized teams
Team A1 and A2 are more effective because of team B's platform



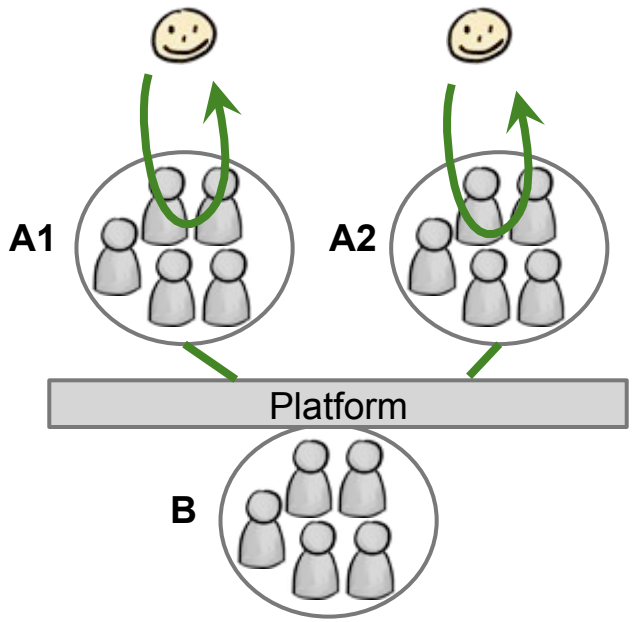
Customer-driven platform teams

External-facing teams

Focus on delivering value to external customers

Internal-facing teams

Focus on making other teams more effective at delivering value to their customers.

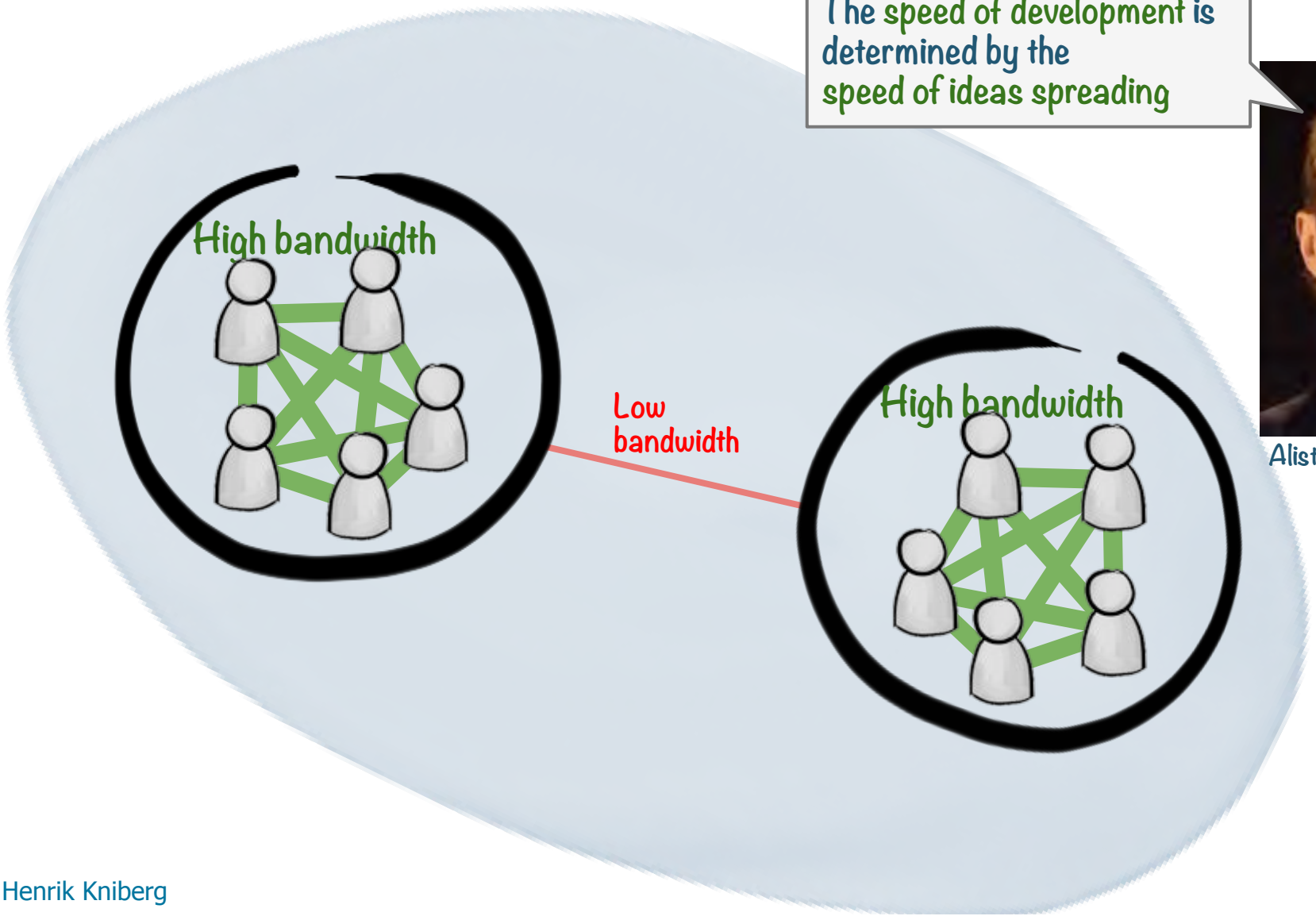


~~The other teams must obey us!~~

The other teams are our customers!

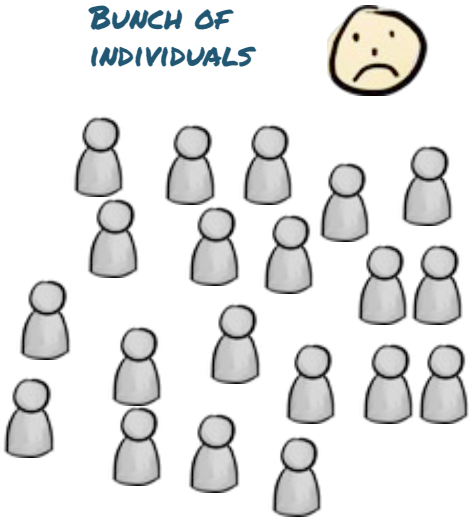
Key decision:

Where can we accept low-bandwidth communication?

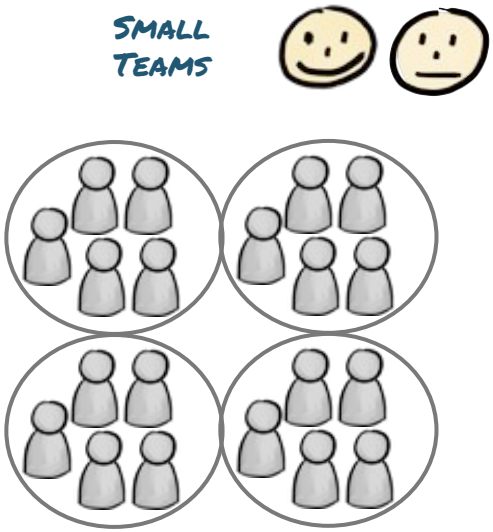


Teams of Teams!

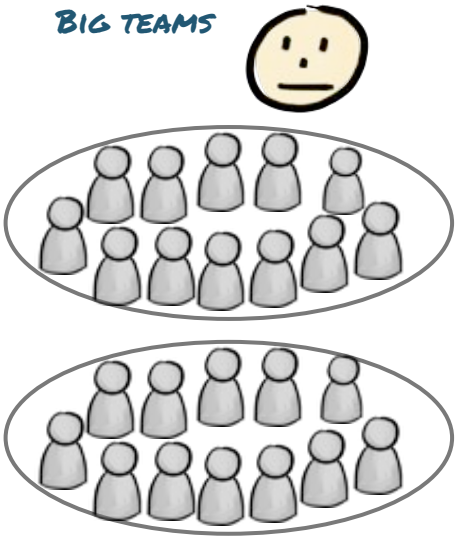
Bunch of individuals



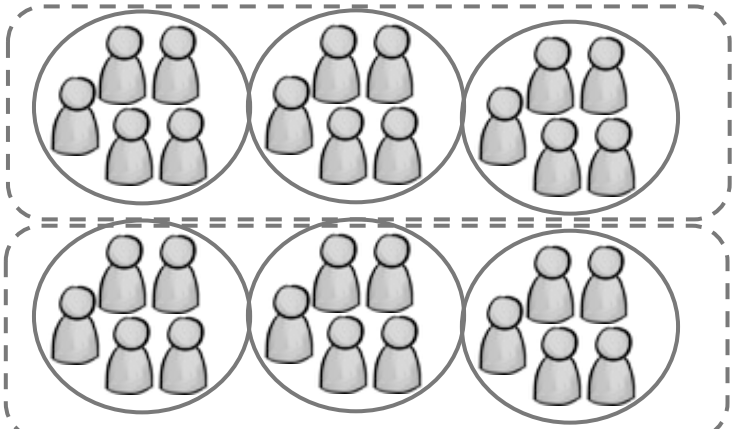
Small Teams



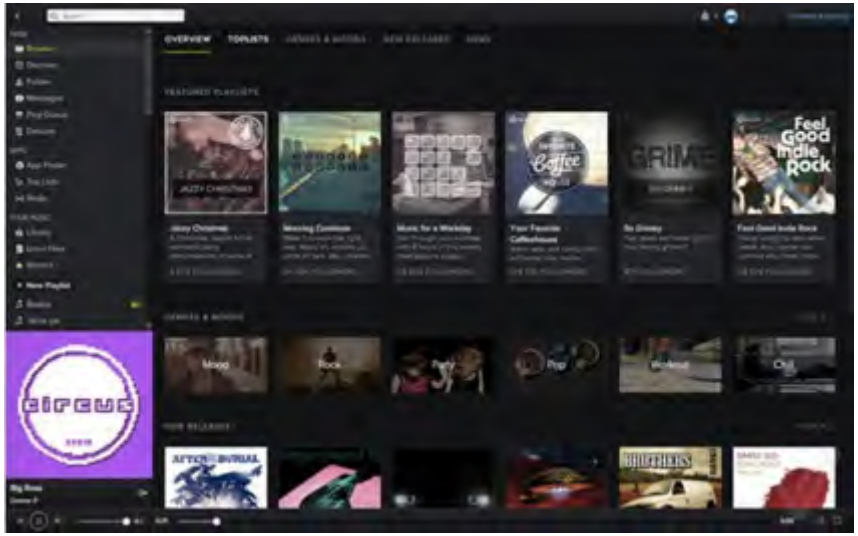
Big teams

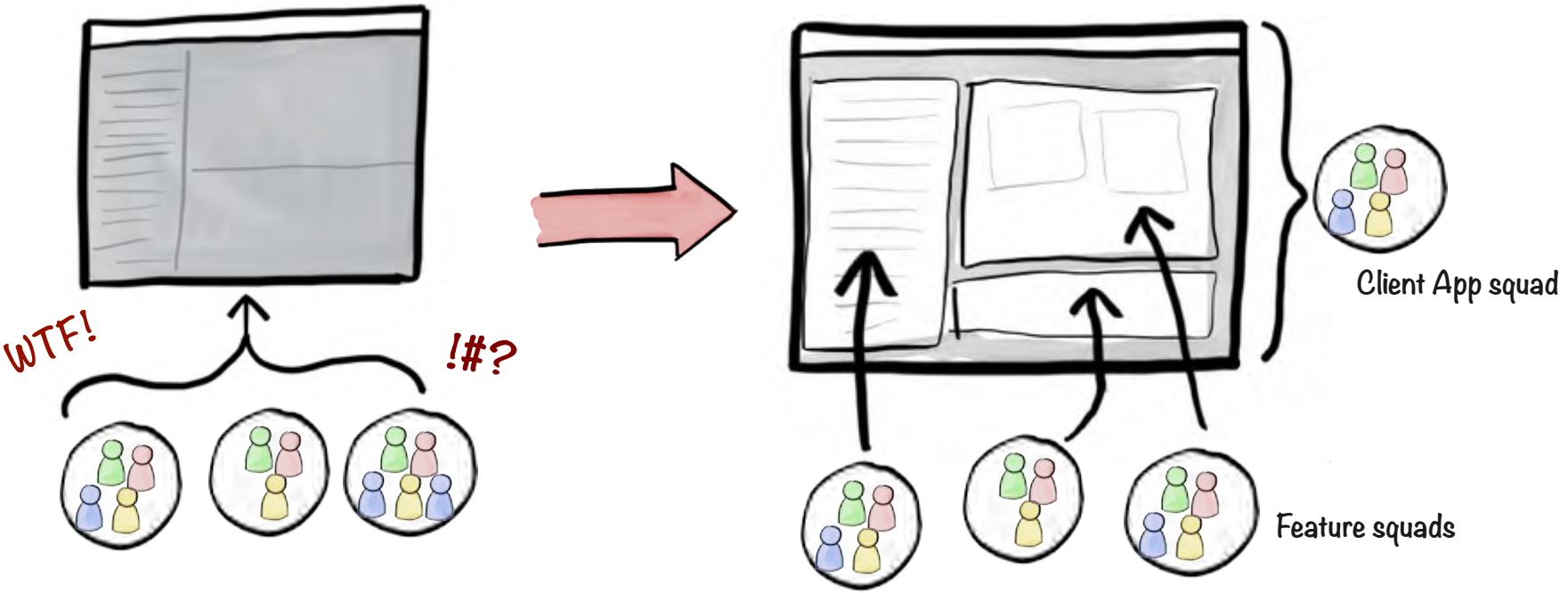


Teams of Small teams



Decoupling to enable frequent releases





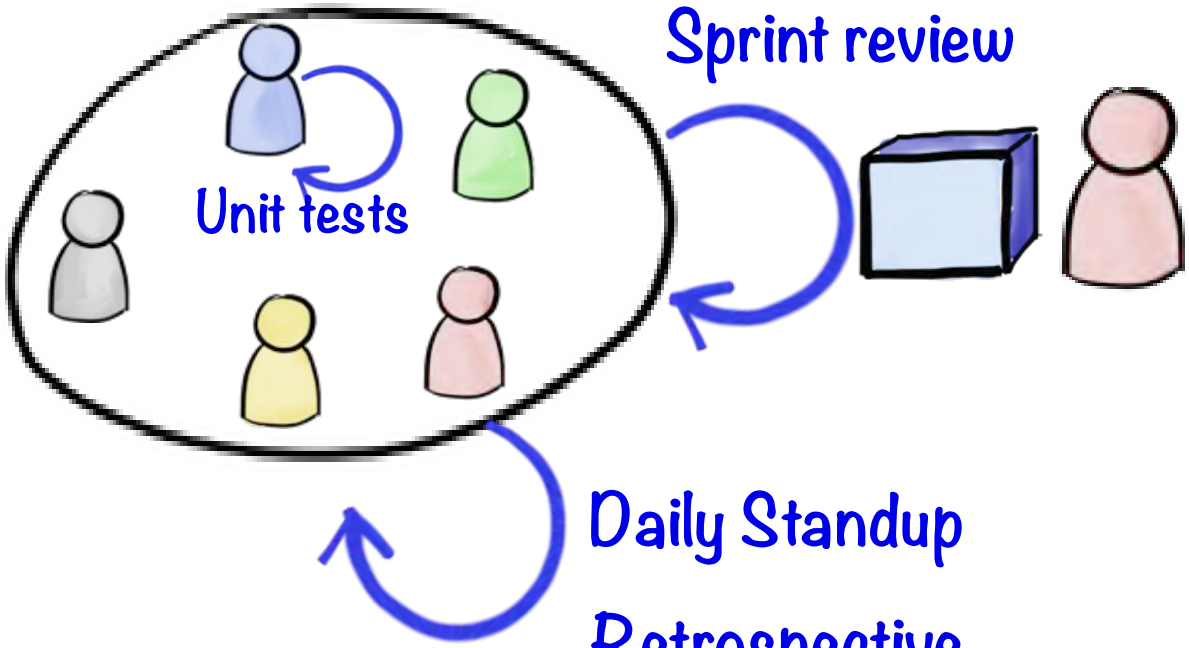
Guidelines for team structure

Try to ensure that each team:

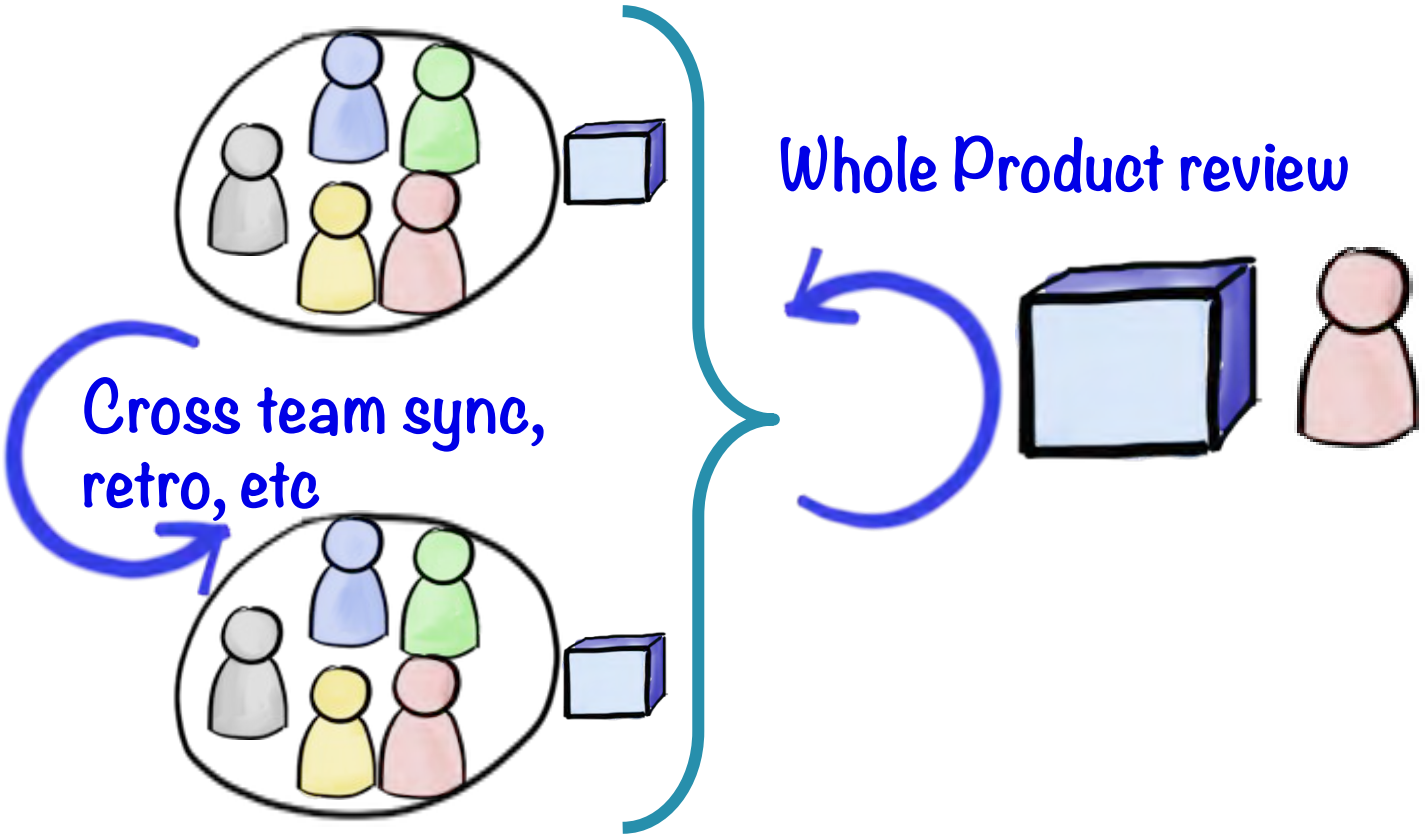
- ❏ is 3-9 people
- ❏ is stable(ish), full-time & co-located.
- ❏ has a mission
- ❏ has clear customers
- ❏ can prioritize between customers
  (ex: via a PO role, or via clear strategic guidelines)
- ❏ cross-functional: has all skills and tools needed to deliver value to customers
- ❏ autonomous: doesn't get blocked waiting for other teams and individuals.

Scale the feedback loop

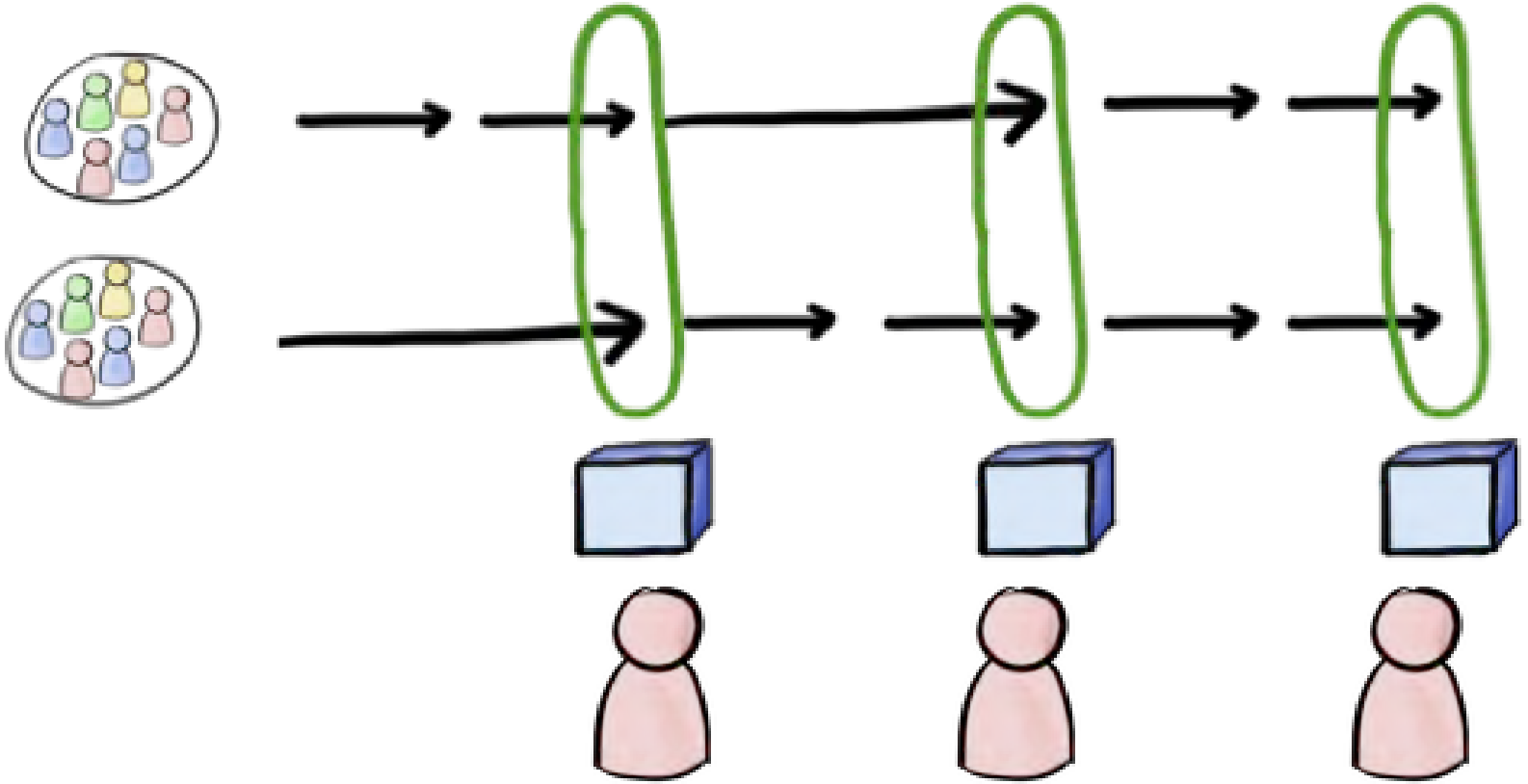
Single team feedback loops



Multiteam feedback loops

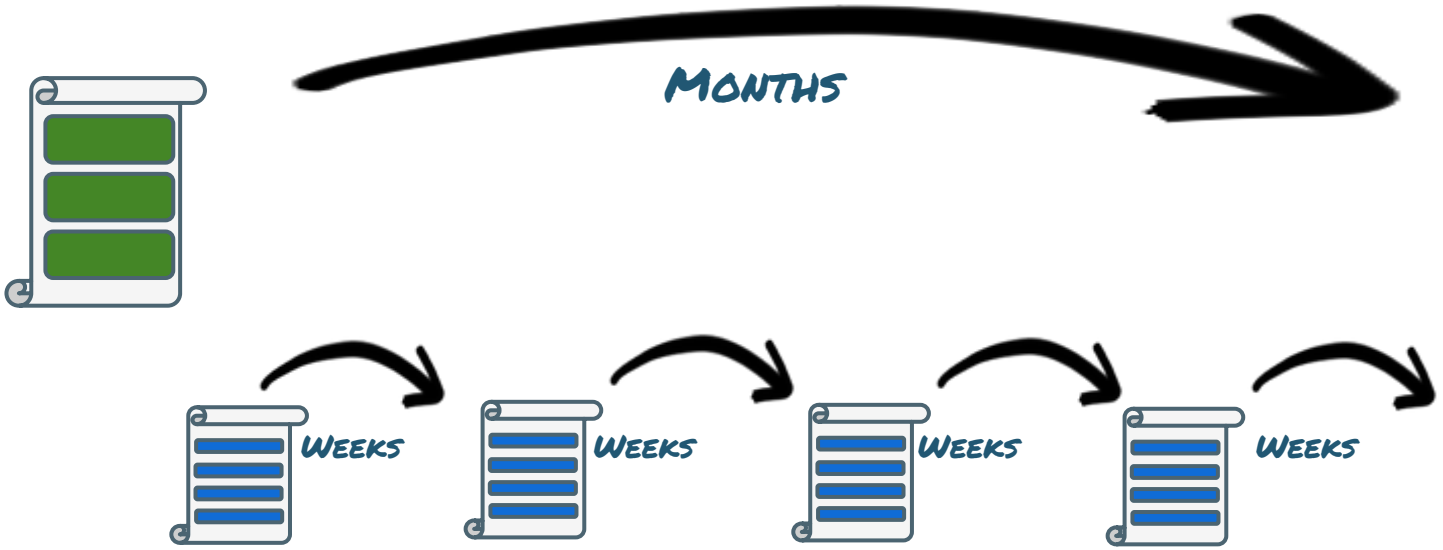


Pattern: Integration Cadence

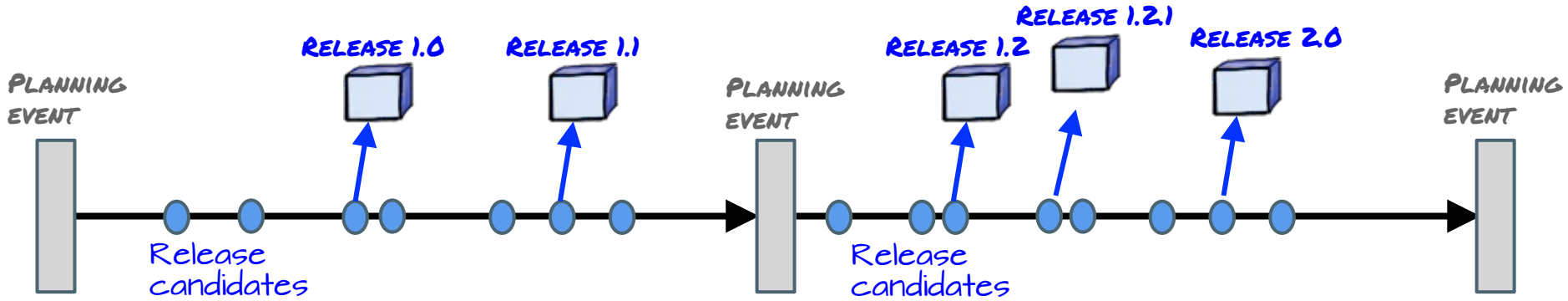


Henrik Kniberg

Pattern: 2-tier planning/alignment

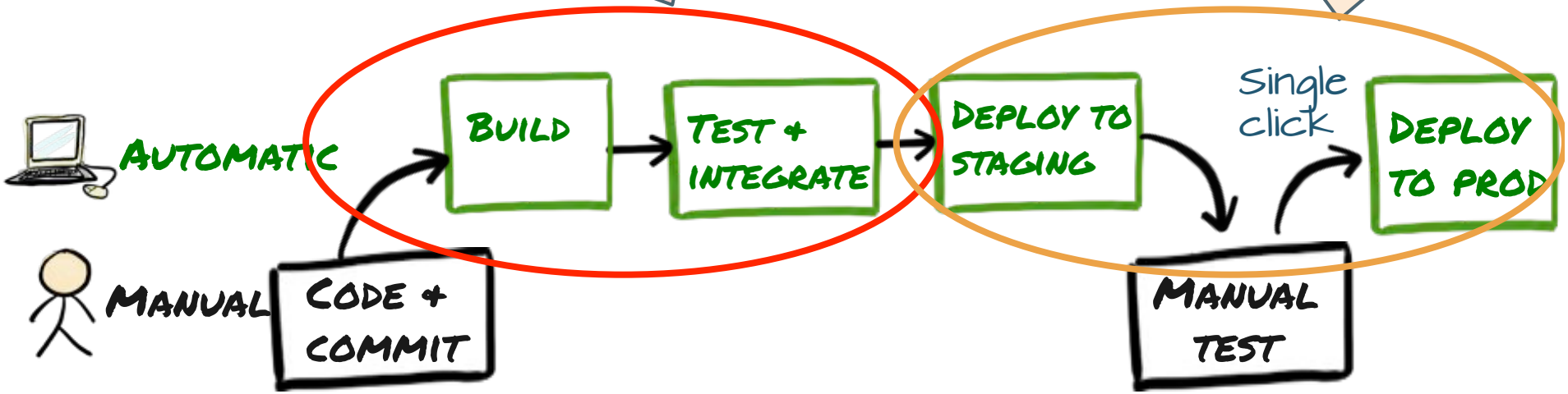


Pattern: Plan on a cadence, release on demand



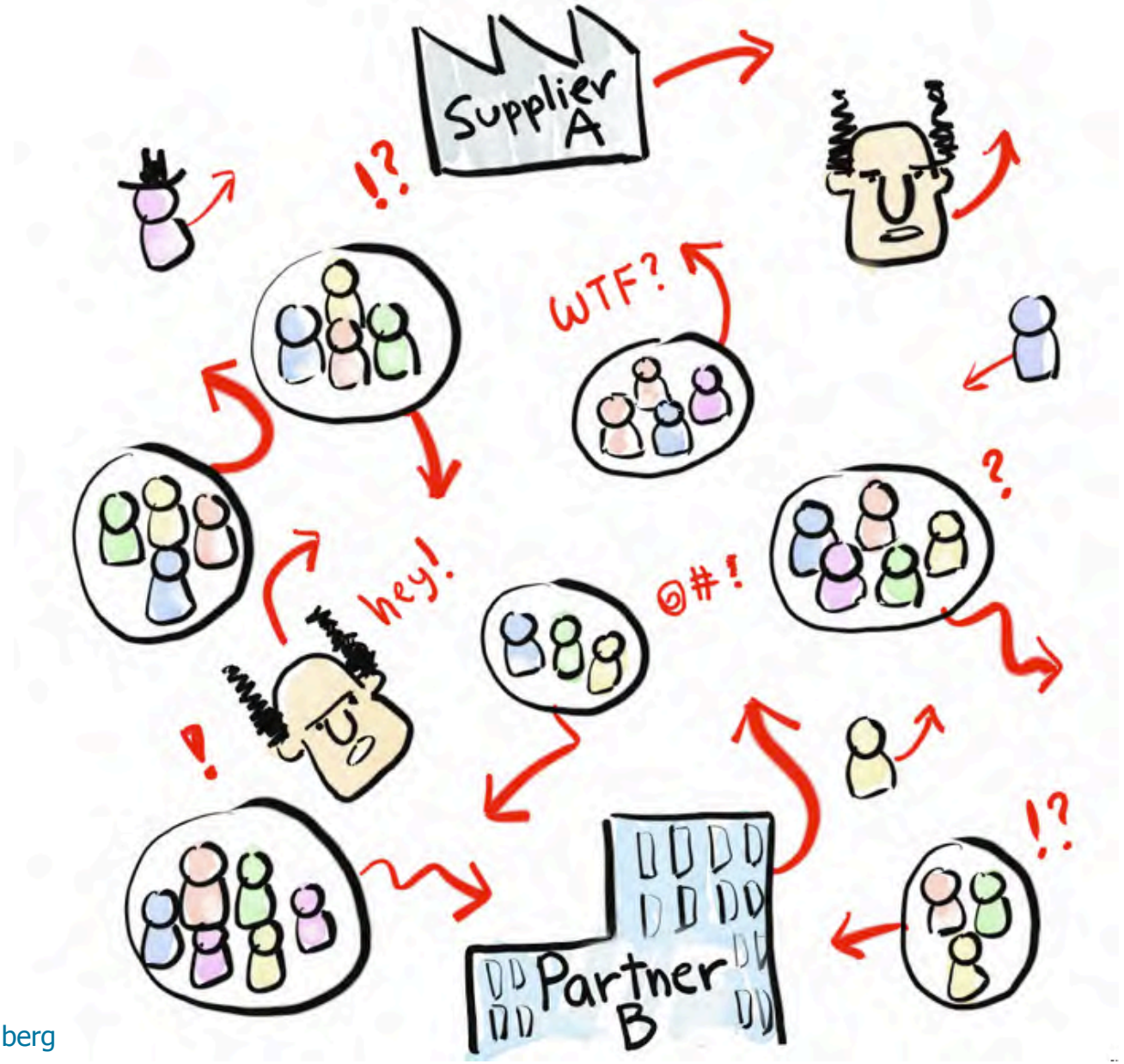
Continuous Integration = Mandatory!

Continuous Delivery = Aspirational.



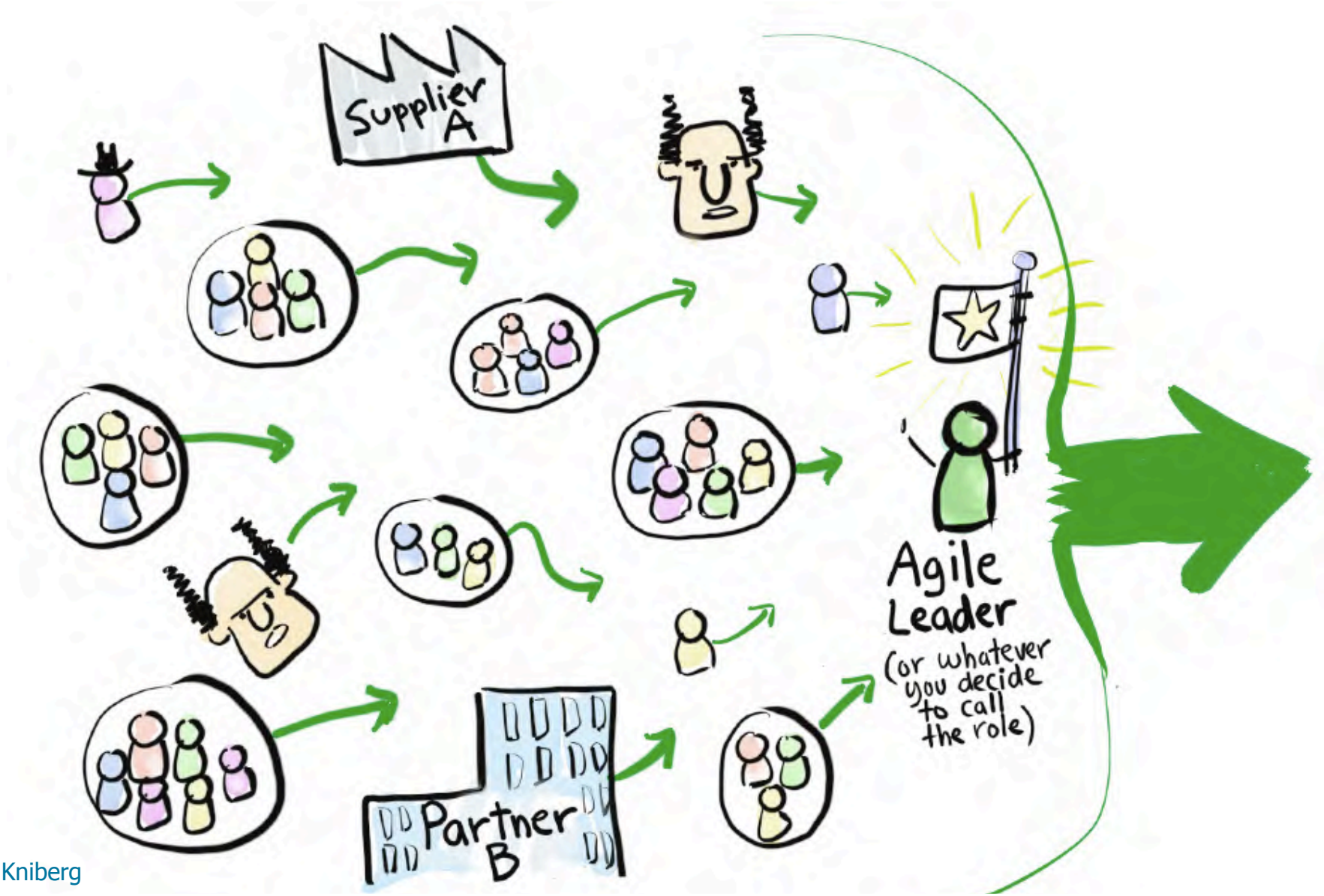
Get everyone aligned

Misaligned teams move very slowly



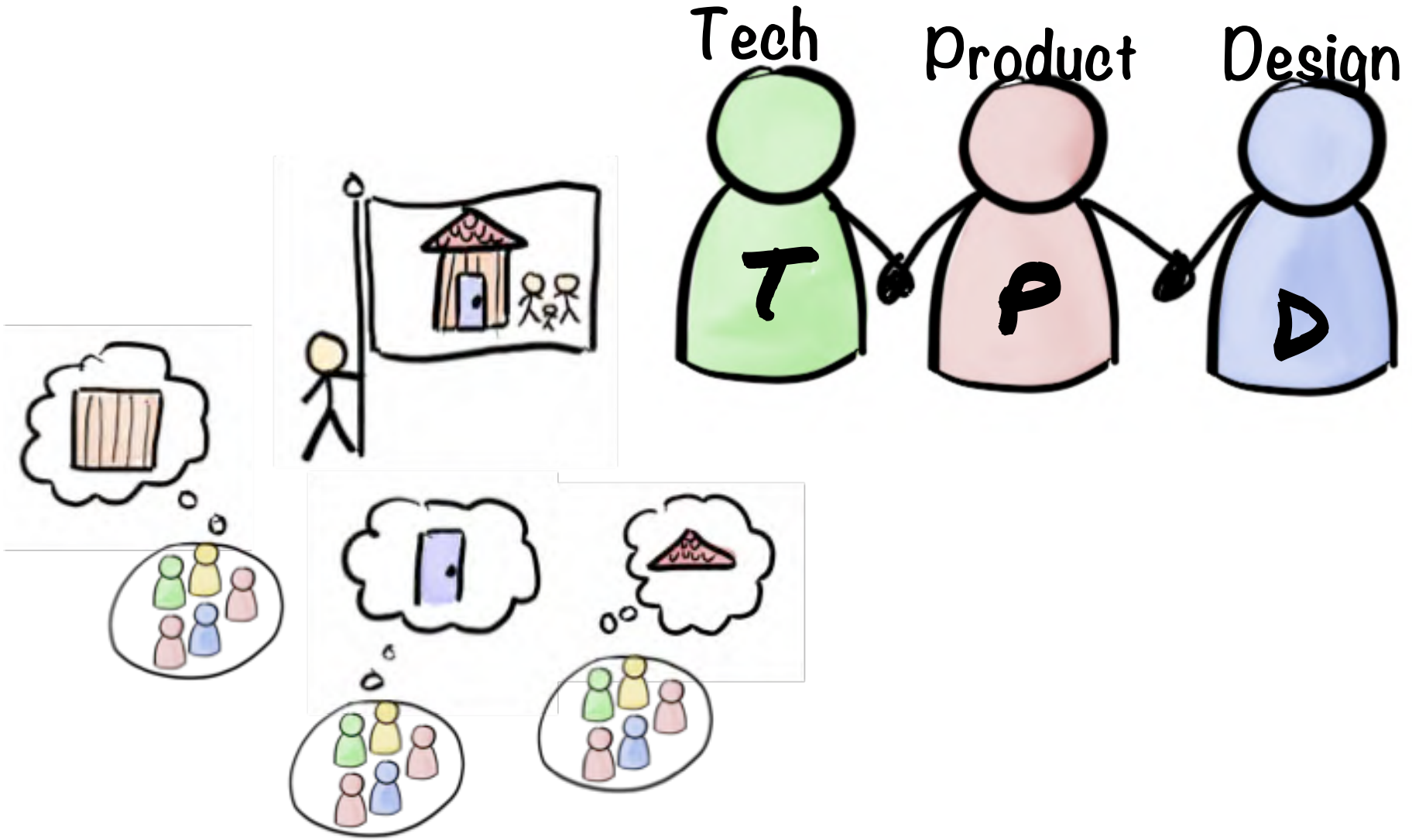
Henrik Kniberg

More teams = more likely that you will need dedicated leader(s)

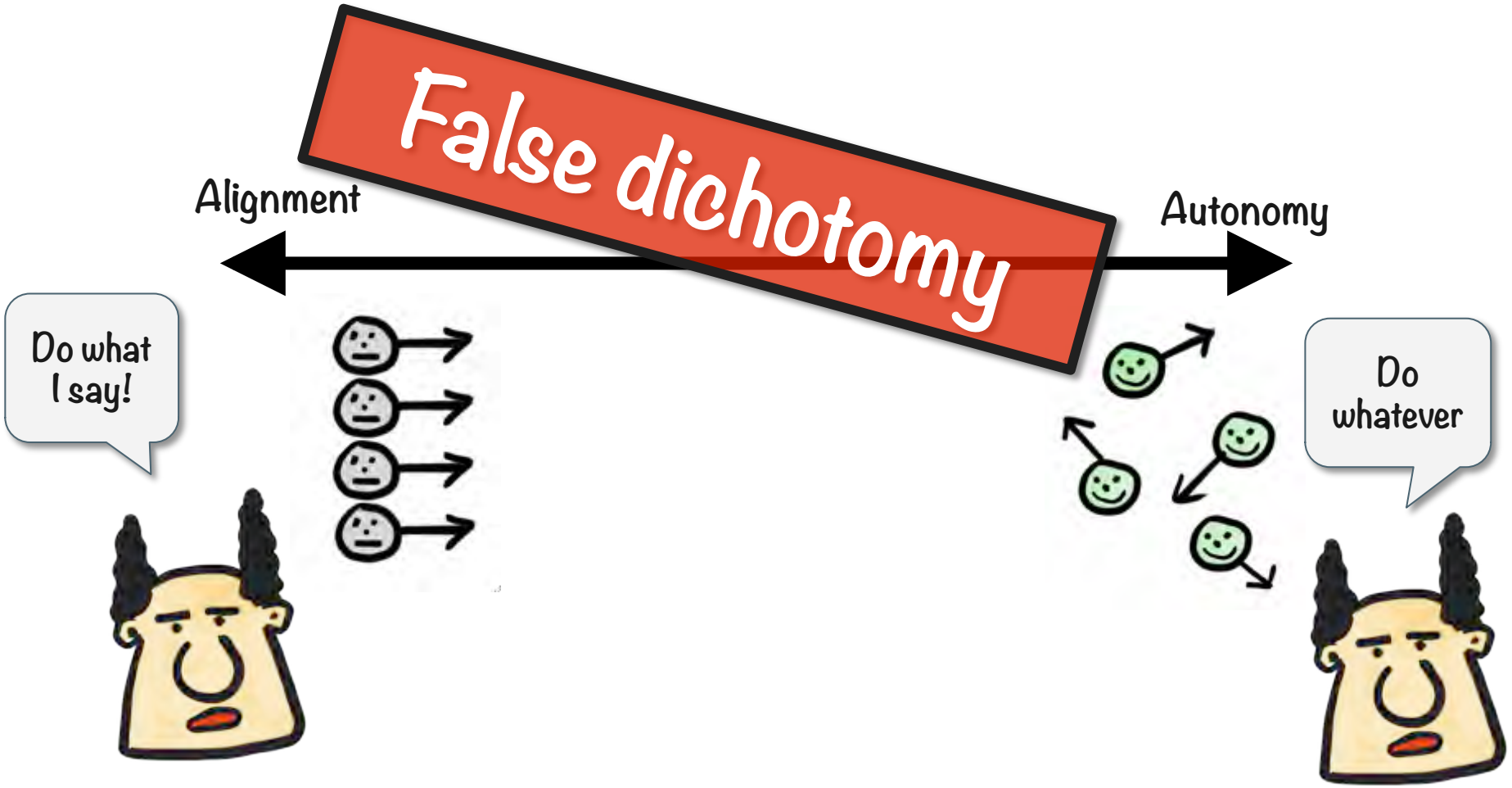


Henrik Kniberg

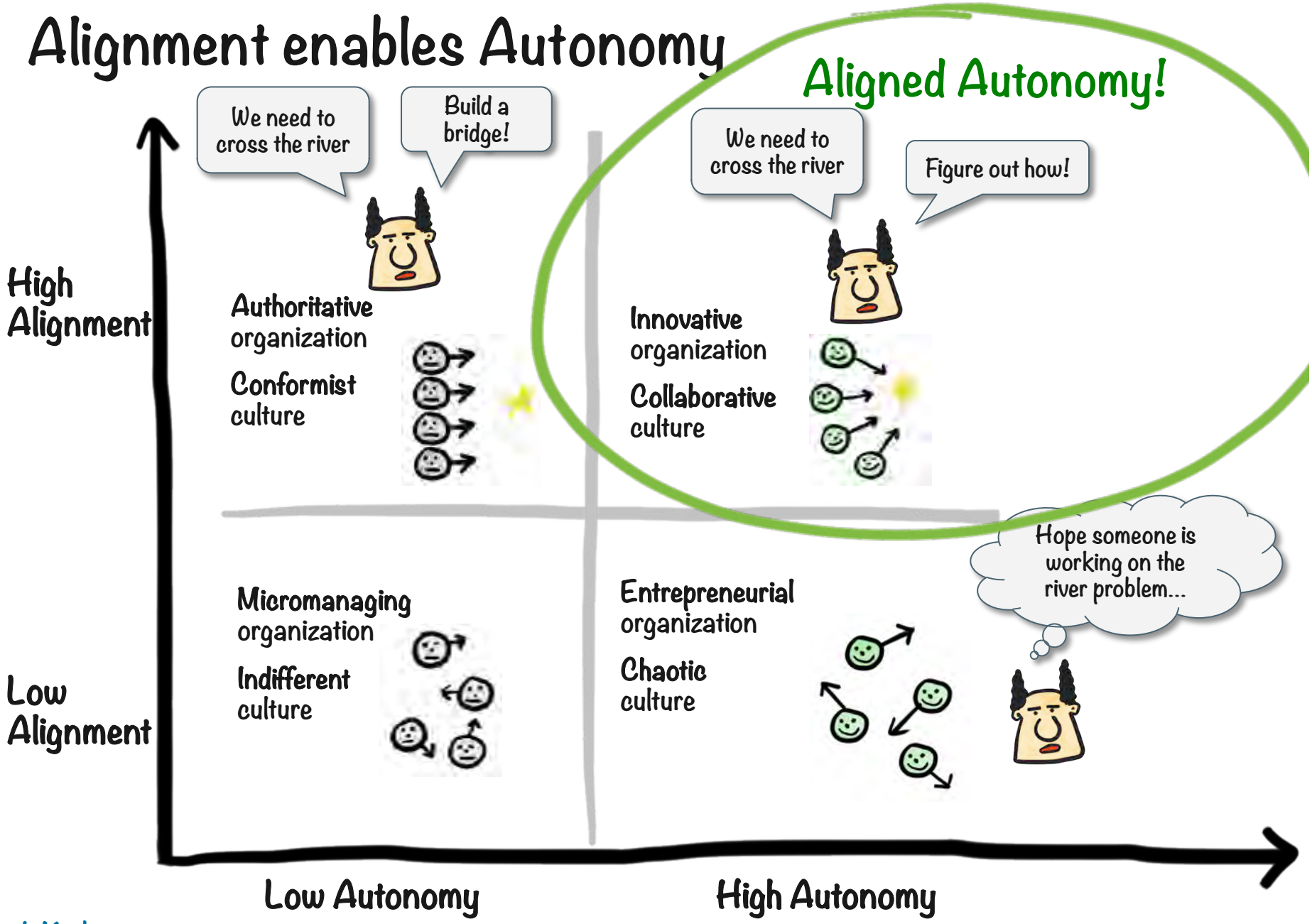
Example: Leadership "Trios"



Alignment & Autonomy



Alignment enables Autonomy



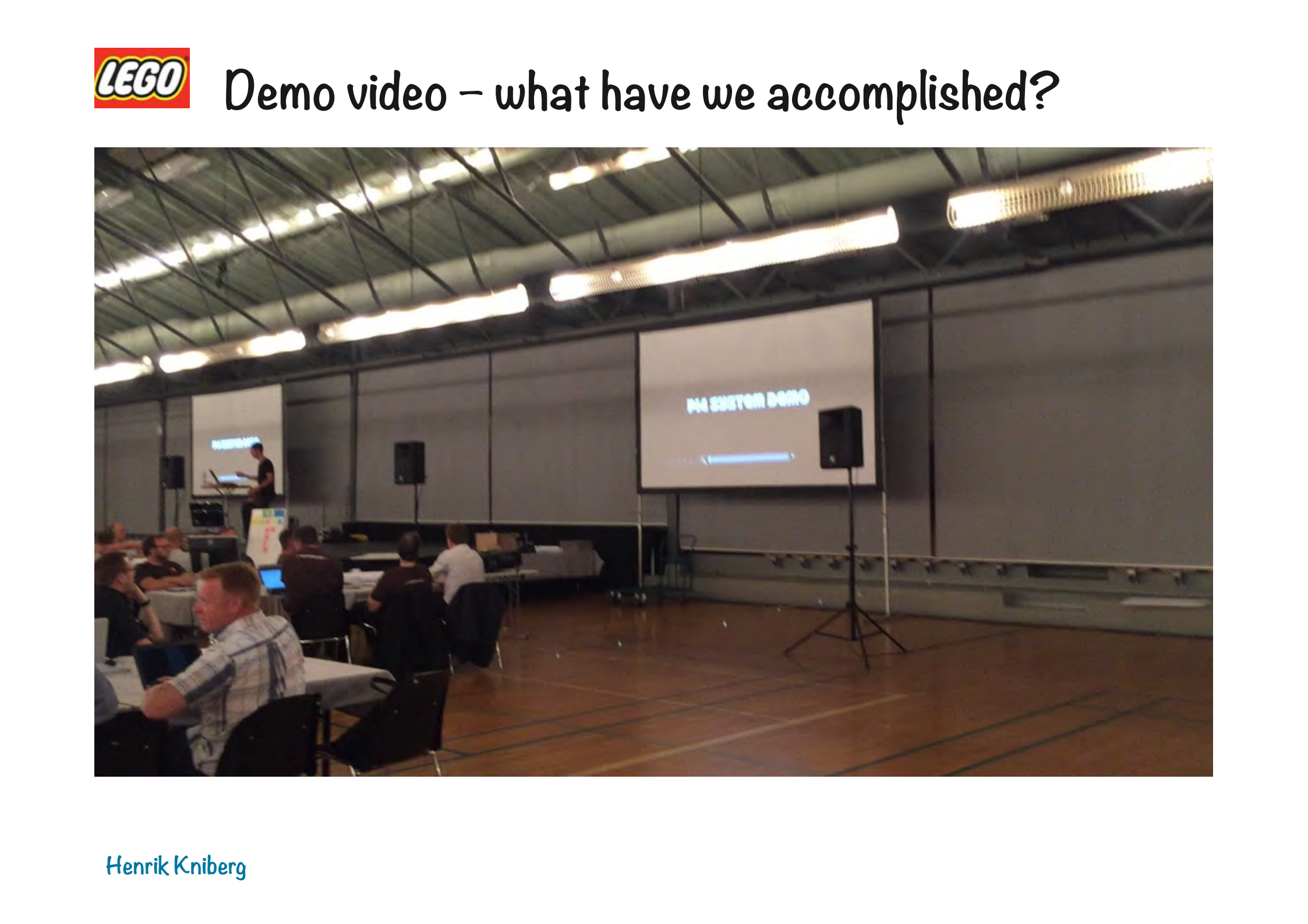
Henrik Kniberg

Example:
Big-room planning/alignment at Lego

Planning as a social event

Henrik Kniberg

2 days, 19 teams, 150 people

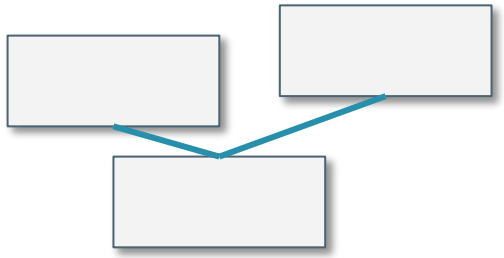


Demo video – what have we accomplished?

Lightning talks

High level priorities:
1. ...
2. ...
3. .....

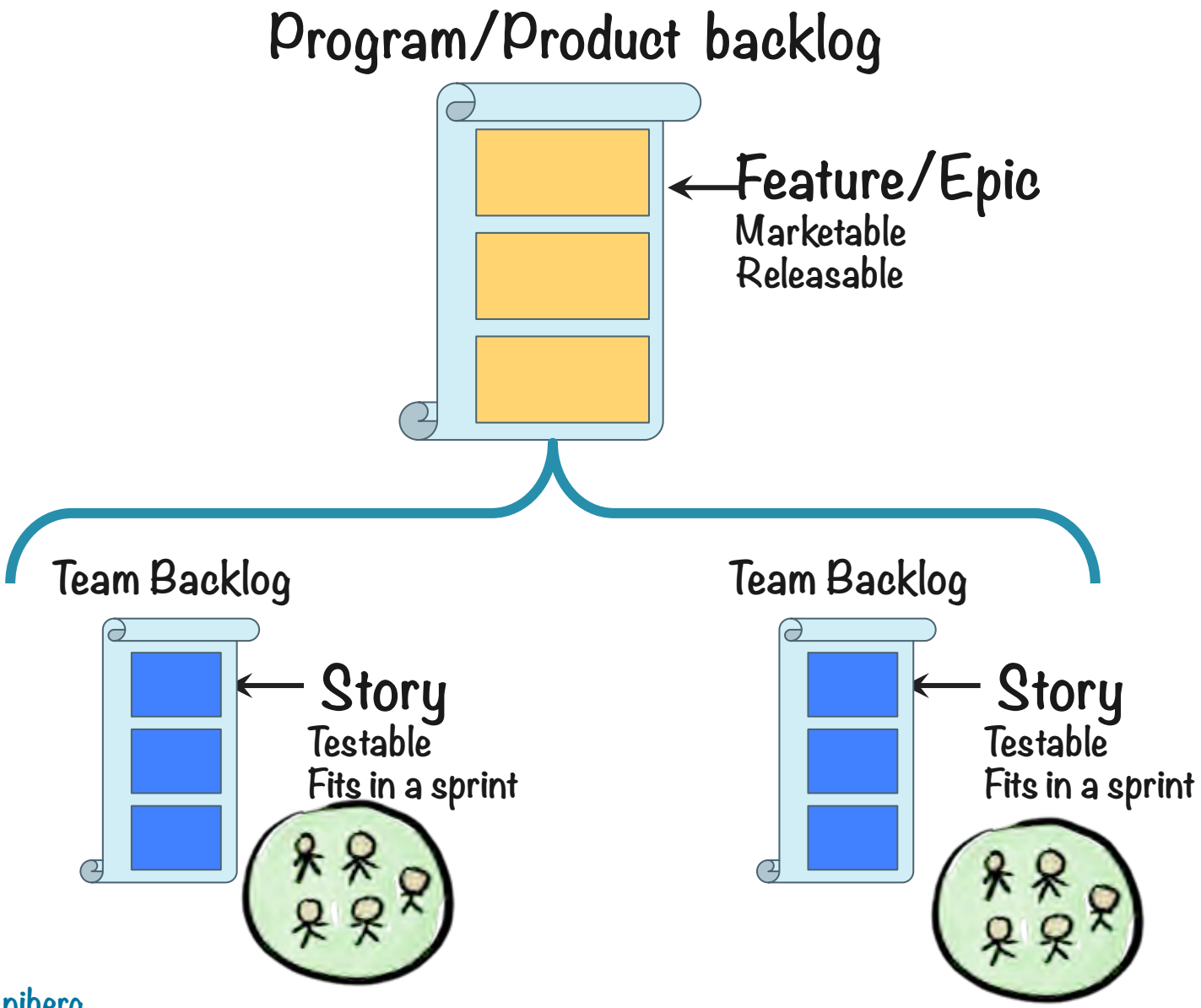
Architecture vision / priorities / constraints



Mission	Inspire and develop the builders of tomorrow	
Aspiration	Globalize and innovate the LEGO system-in-play	
Promises	**Play Promise** Joy of building. Pride of creation	**Partner Promise** Mutual value creation
	Planet Promise Positive impact	**People Promise** Suceed together
Spirit	Only the best is good enough	
Values	Imagination - Creativity - Fun - Learning - Caring - Quality	

Henrik Kniberg

Pattern: Different levels of granularity

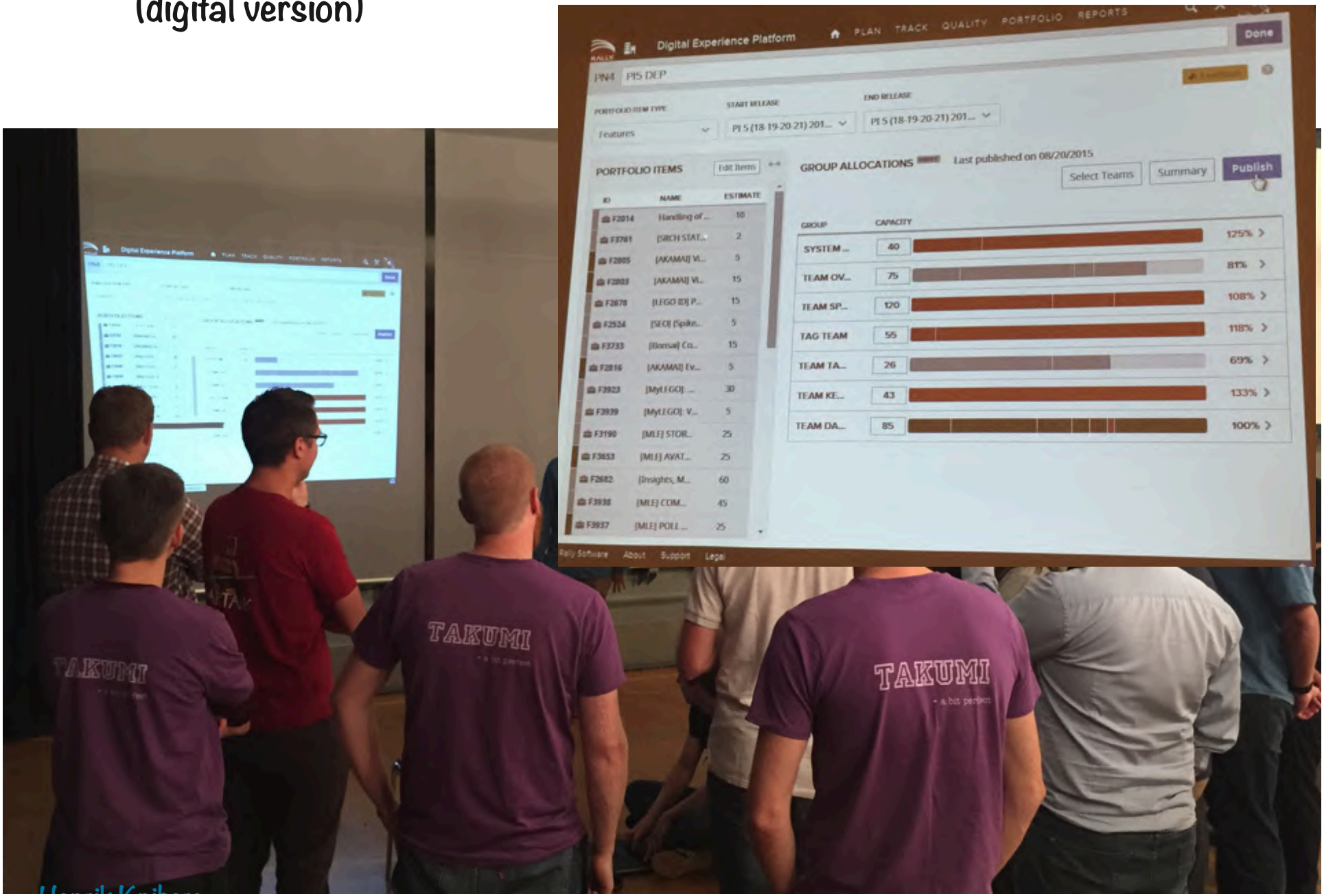


Henrik Kniberg

Team breakout: Pulling from the program backlog

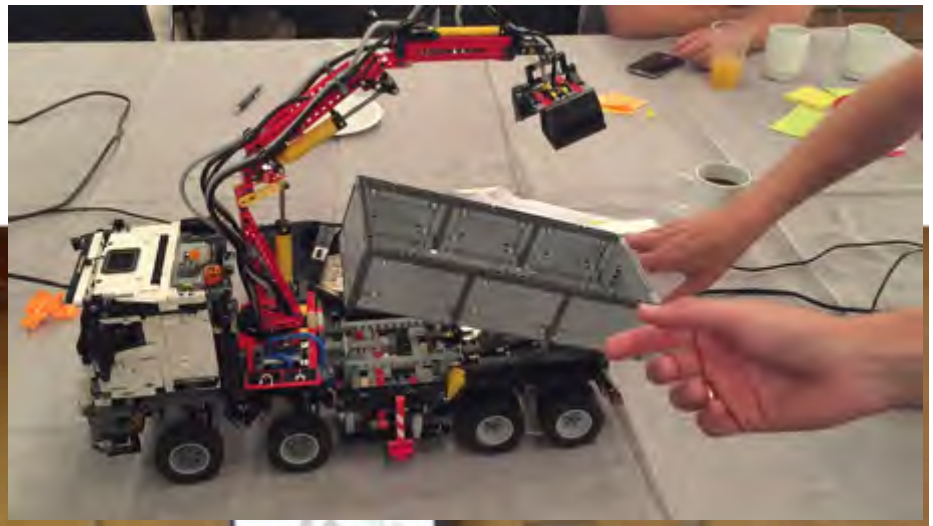
Henrik Kniberg

Team breakout: Pulling from the program backlog
(digital version)



Team breakouts

Law of 2 feet....



Henrik Kniberg


LEGO
Program Board
(a.k.a Dependency Board)
Henrik Kniberg

Early detection of dependency problems

Henrik Kniberg

Scrum of Scrums = dependency sync

Simpler version of dependency sync

Risk board
(per project/epic)

Henrik Kniberg

Management review / problem solving

Henrik Kniberg

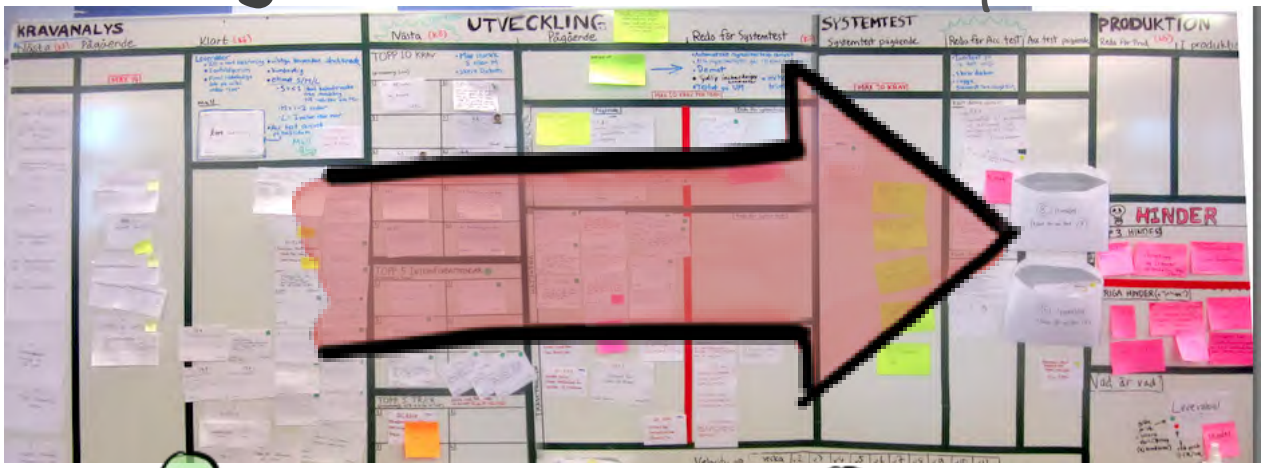
Day 2

Management feedback & commitment to help

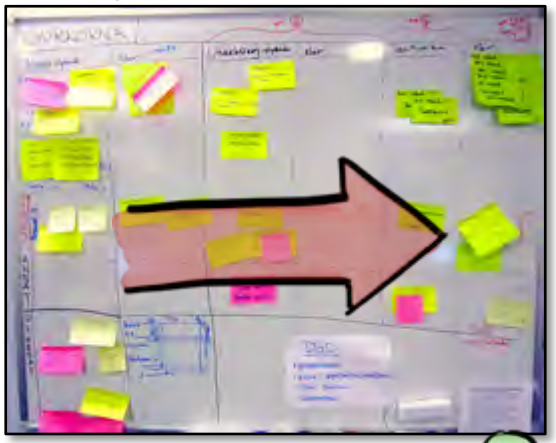
Pattern: Information radiators

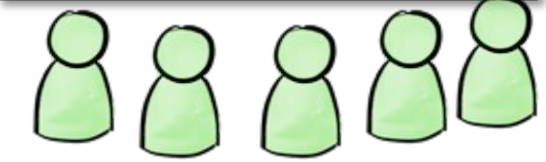
- In the hallway
- Or in a ~~War Room~~ Zen Room

Big Picture – features/epics

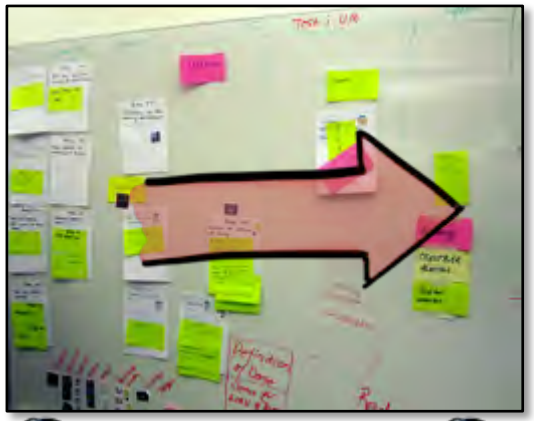


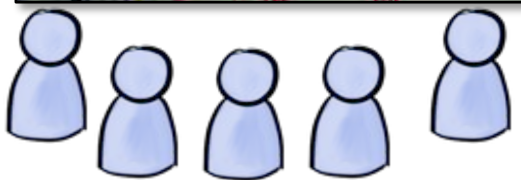
Team 1 - stories



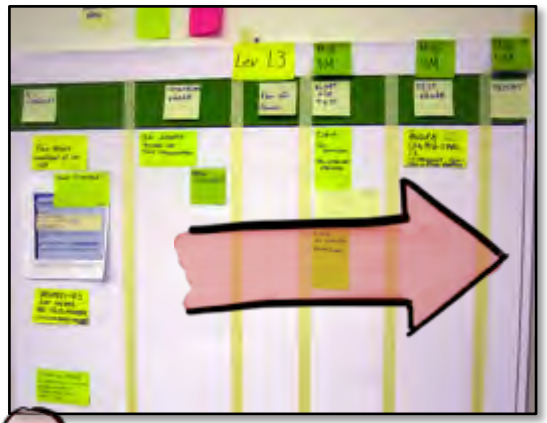


Team 2 - stories





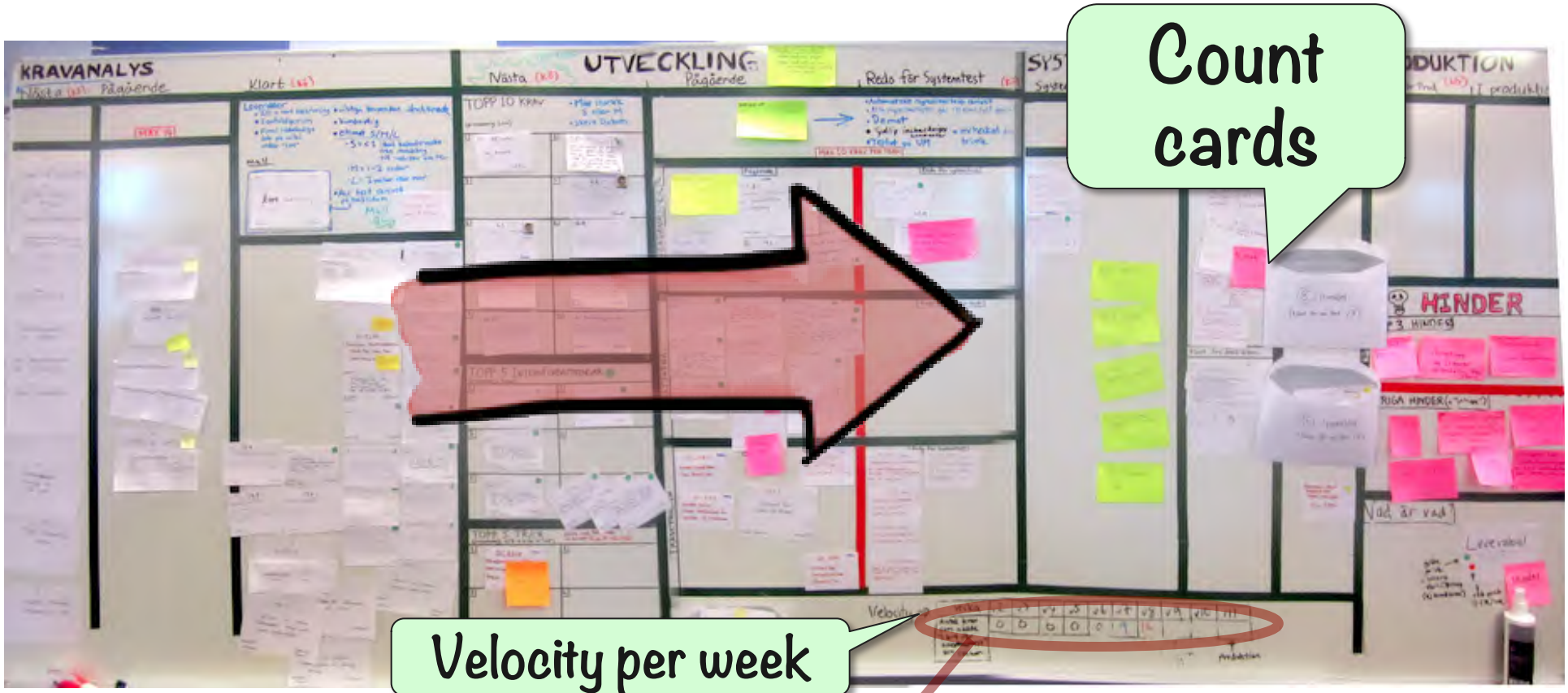
Team 3 - stories



Henrik Kniberg

Polisen

Example: Measuring velocity by counting cards



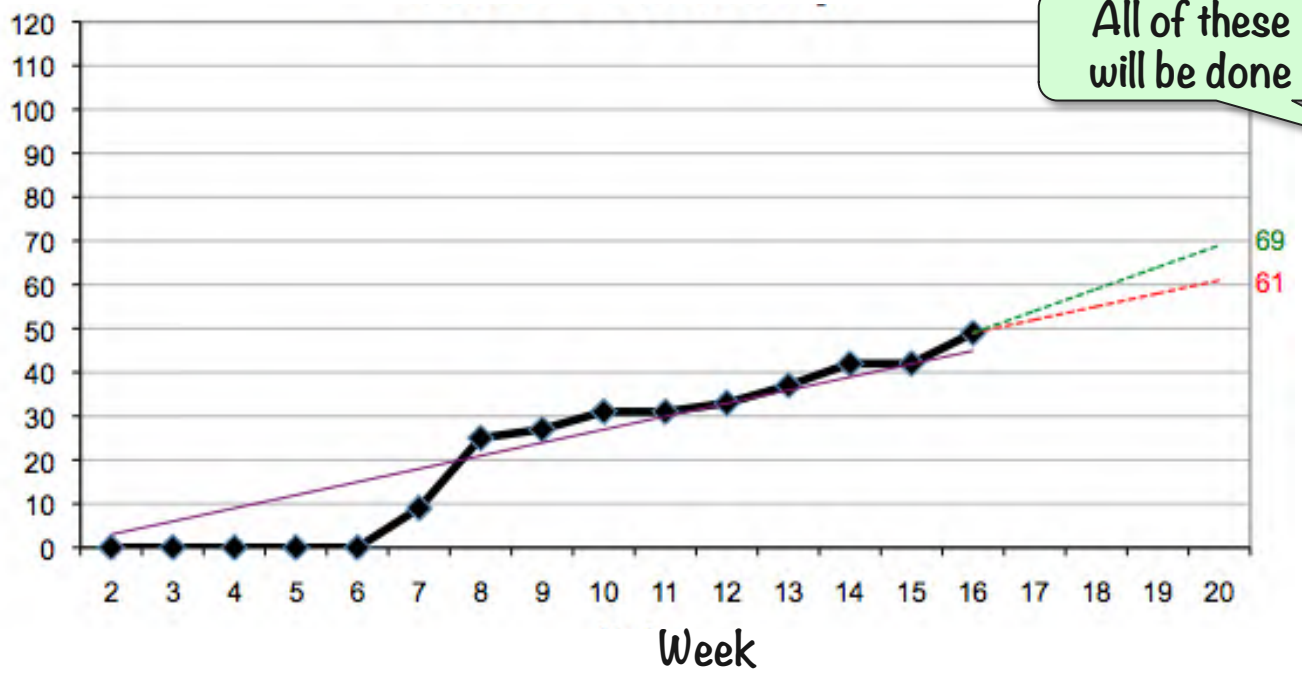
Vecka	v10	v11	v12	v13	v14	v15	v16	v17	v18
Antal nya funktioner som nått till 'Redo för AccTest'	4	0	2	4	5	0			

↑ Prodsättn.

VEL

Polisen

Example: Release planning using a burnup chart



Total # of delivered features

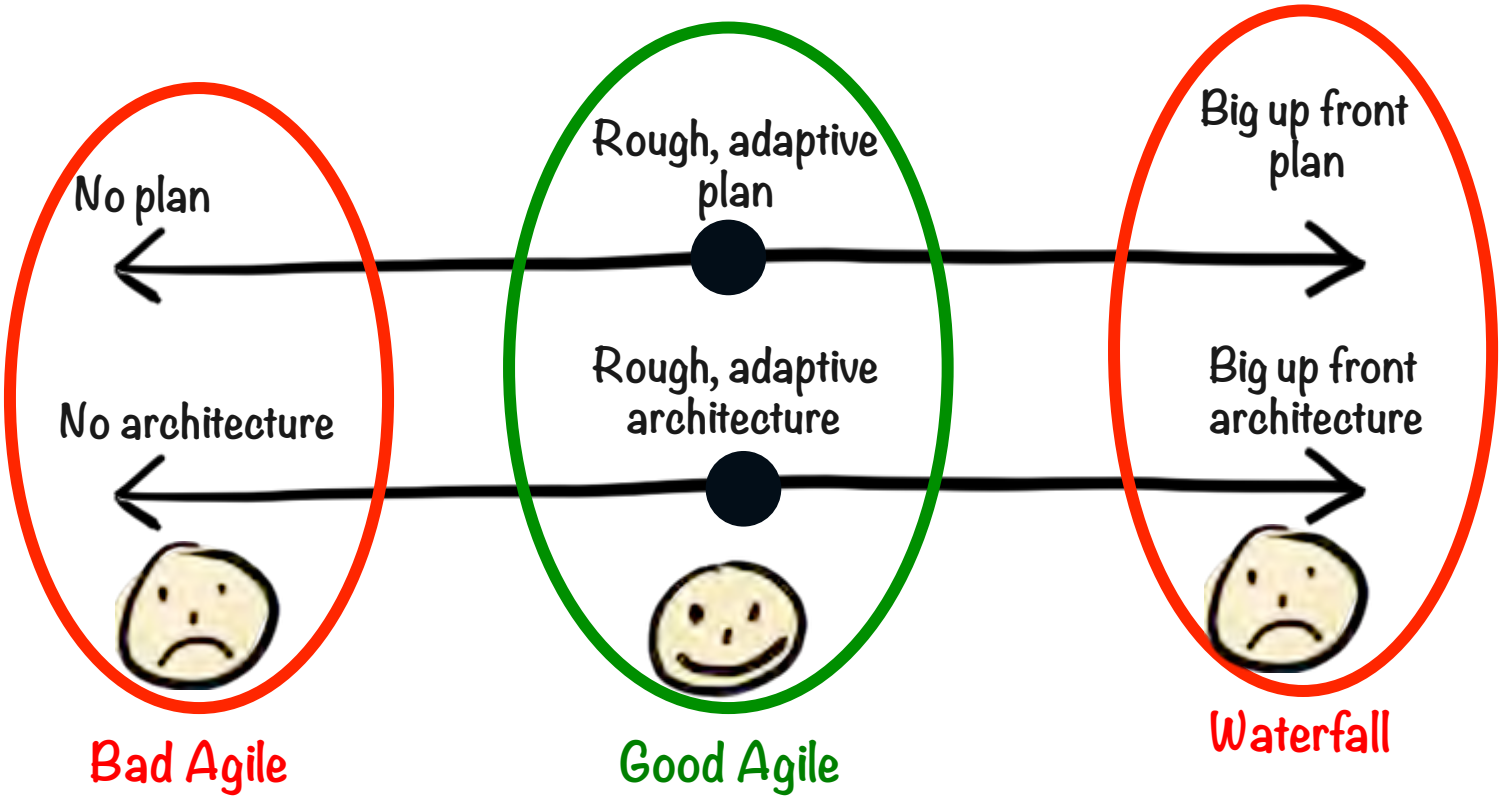
All of these will be done

Some of these will be done, but not all

None of these will be done

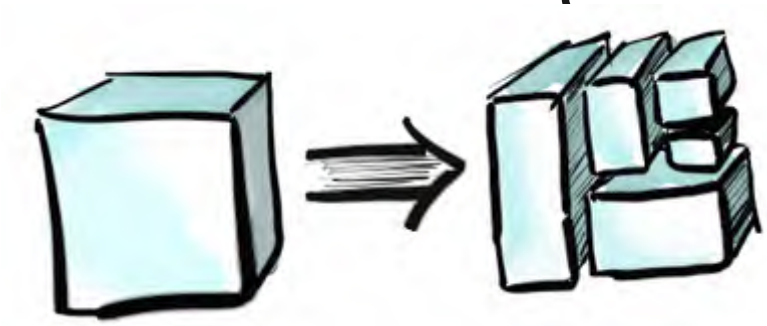
Wrapup

Don't go overboard with Agile!

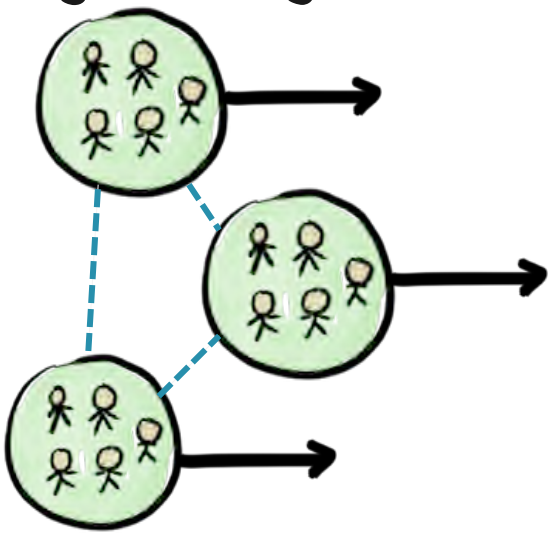


Stuff you need to figure out with multiple teams

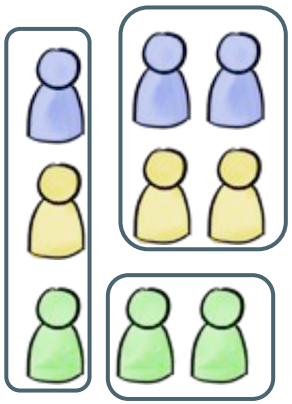
How to slice the elephant



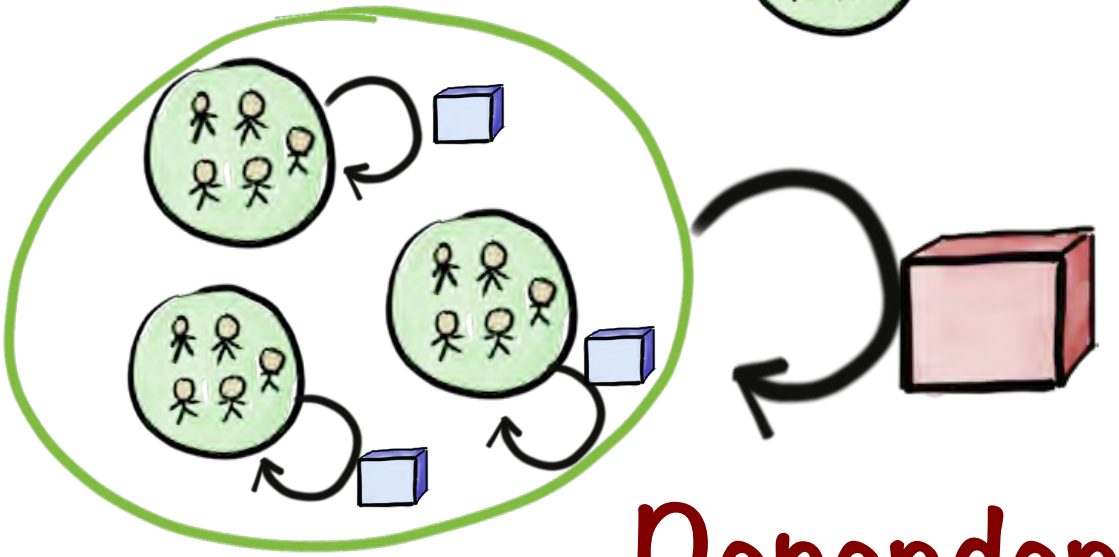
Team sync / alignment



Team structure



Feedback loop



Dependencies!

Henrik Kniberg

Real-life agile scaling – take aways

- **Scaling hurts**
  Keep things as small as possible
- **Agile is a means, not a goal**
  Don't go Agile Jihad. Don't dump old practices that work
- **There is no "right" or "wrong" way**
  Just tradeoffs
- **There is no one-size-fits-all**
  But plenty of good practices
- **Build feedback loops at all levels**
  Gives you better products and a self-improving organization.