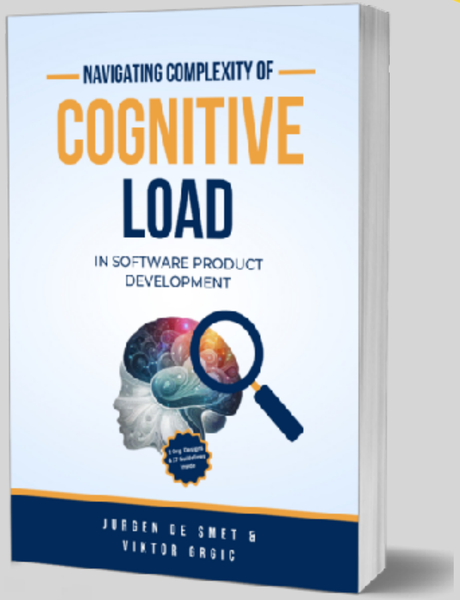
Strategies to Reduce Cognitive Load in Large Organizations



Viktor Grgic @vgrgic, viktor@odd-e.com, Odd-e Netherlands B.V. odd-e.com less.works

Definition of Cognitive Load

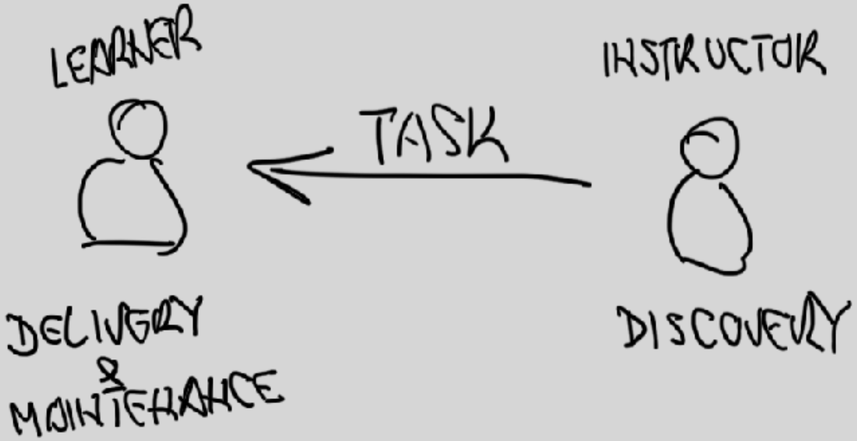
Amount of working memory resources used.

- **Intrinsic Cognitive Load:** The effort needed to change the product.
- **Extraneous Cognitive Load:** The effort needed to read and understand what to change in a product.
- **Germane Cognitive Load:** The effort needed to develop permanent knowledge, or, in other words, expertise.

Extraneous Cognitive Load

Make distinction between

- Presentation of information.
- Presentation of a task.



High **extraneous** cognitive load

Low **germane** cognitive load

Demands more routine expertise

(to be replaced by AI in future)

Focus on Adaptive expertise

Low **extraneous** cognitive load

High **germane** cognitive load

Myths

Myth: “Teams are cognitively overloaded because they must know so much about components to properly maintain them.”

Myth: “A team feels cognitively overloaded if they have too many responsibilities to handle”

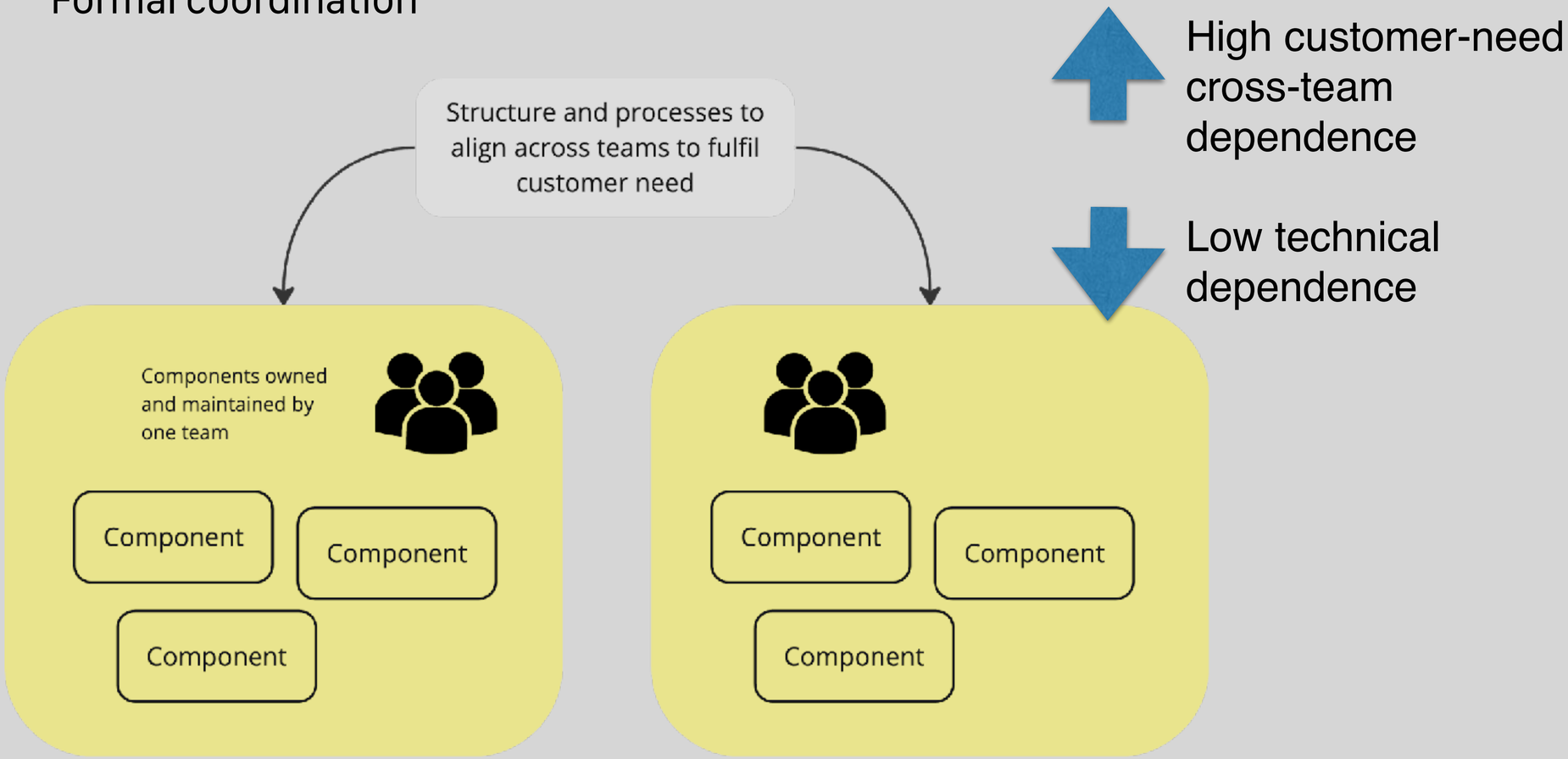
Myth: “Let developers do only development and maybe testing for better focus”

Common organizational structures and issues

Viktor Grgic @vgrgic, viktor@odd-e.com, Odd-e Netherlands B.V. odd-e.com less.works

Pattern 1: High autonomy in building

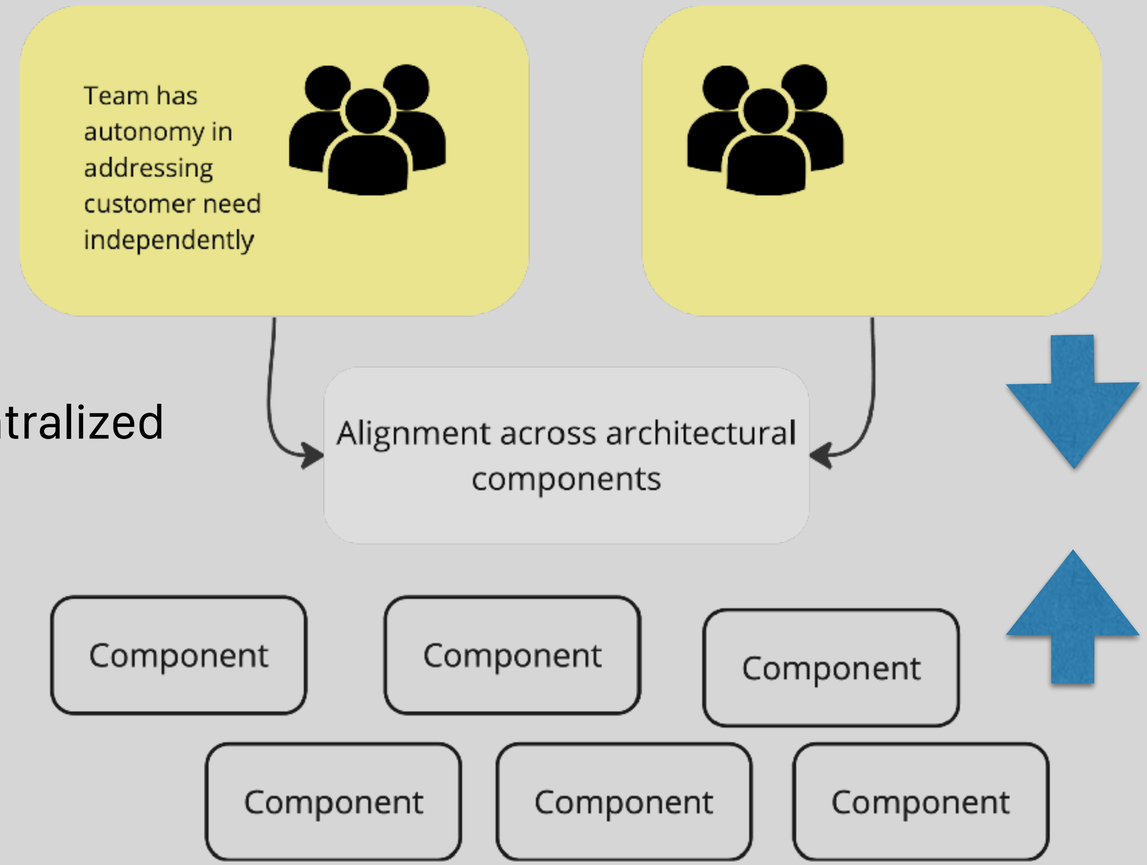
Formal coordination



High customer-need cross-team dependence

Low technical dependence

Pattern 2: High autonomy in solving customer problems

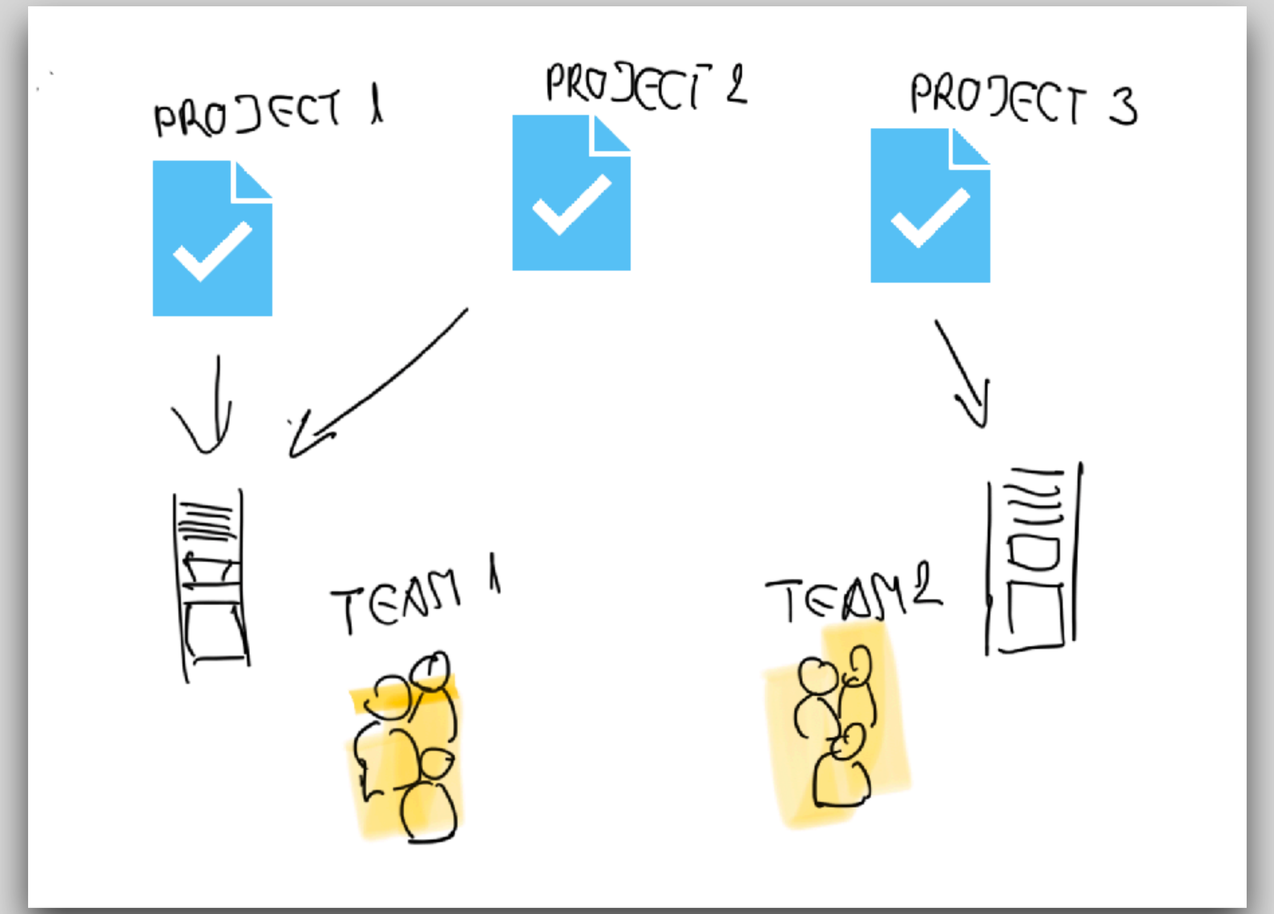


Informal / decentralized coordination

Low customer need cross-team interdependence

High technical interdependence

Everything is a project

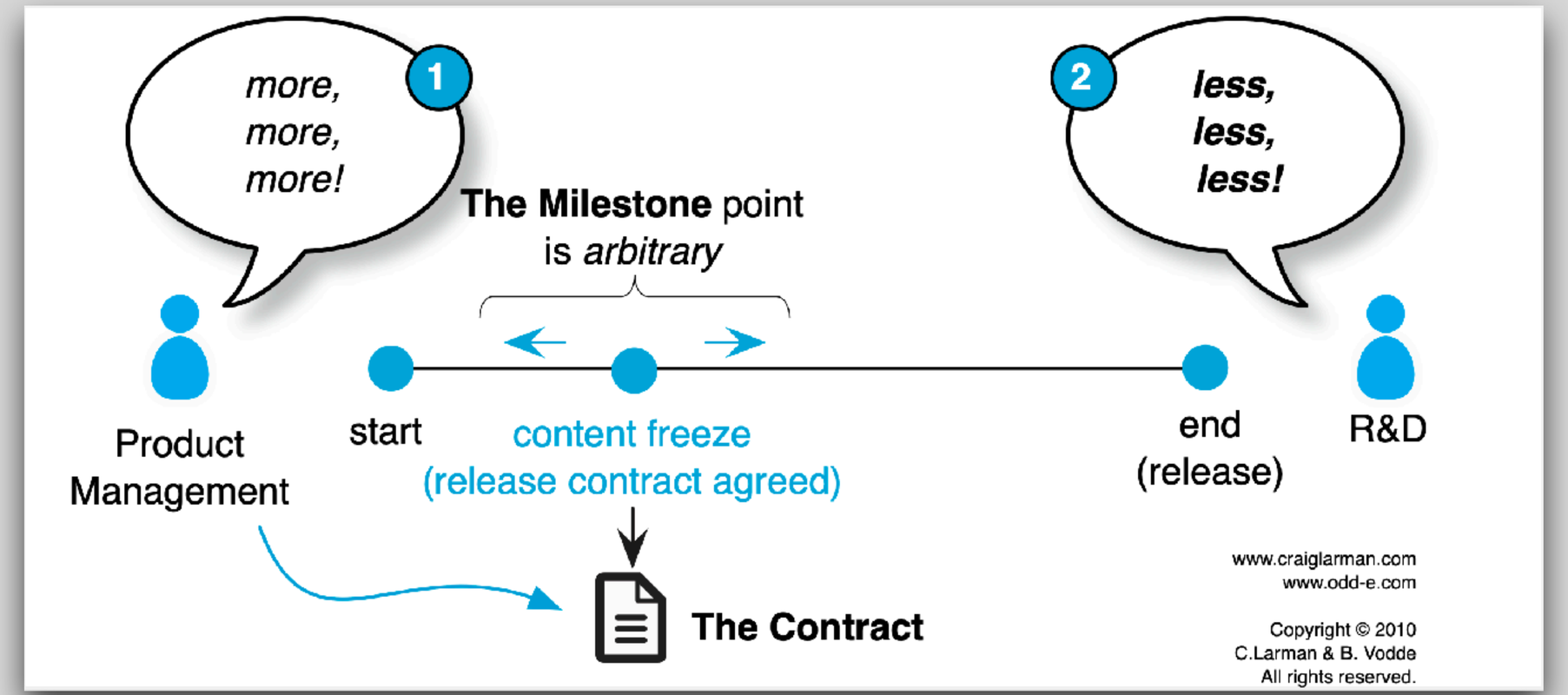


Low autonomy in solving customer problem

Low autonomy in building

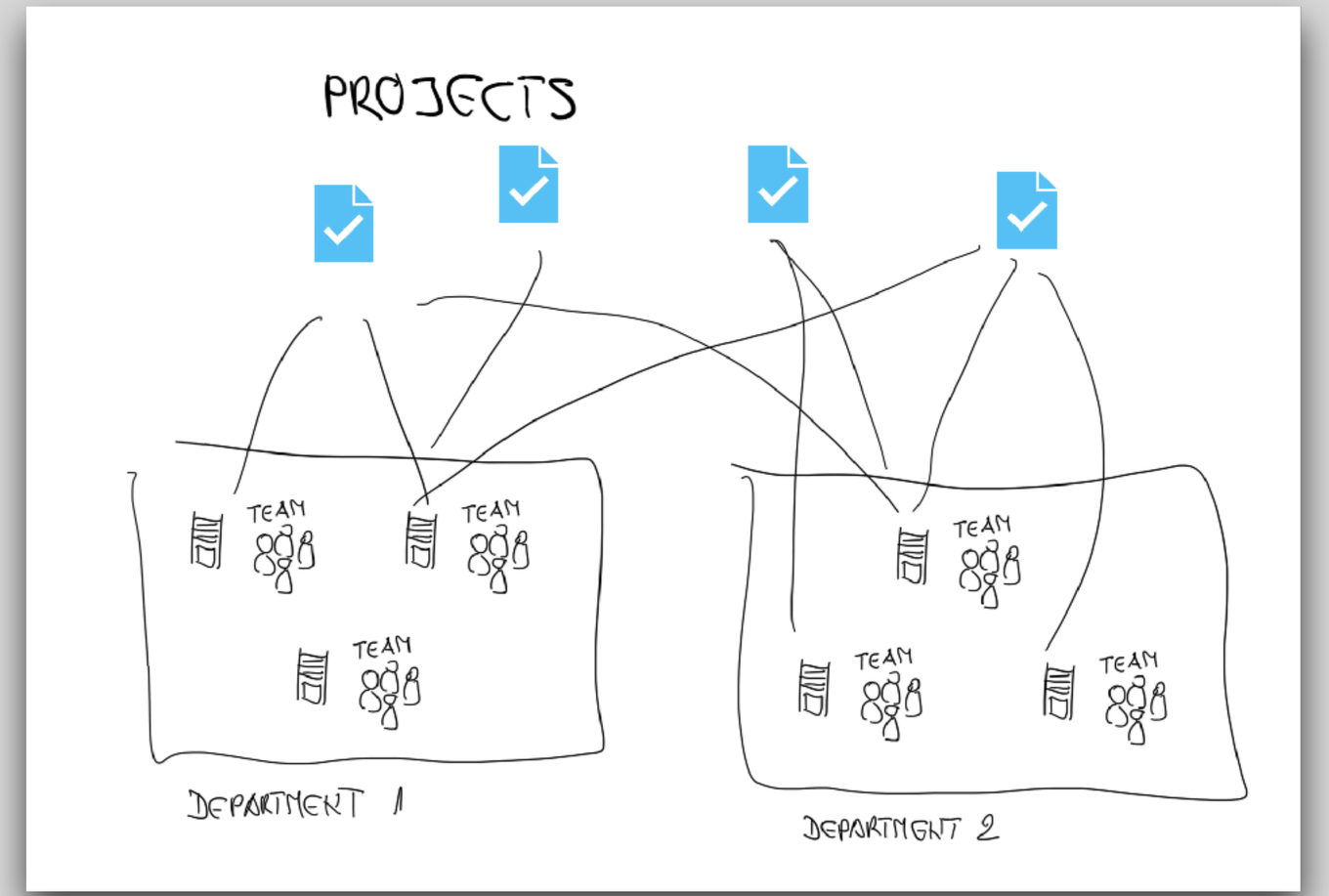
Teams are badly performing temporary resources

Contract game structure



Typical question in case of contract game:
"How do we improve accuracy of estimates?"

Fixed teams in departments

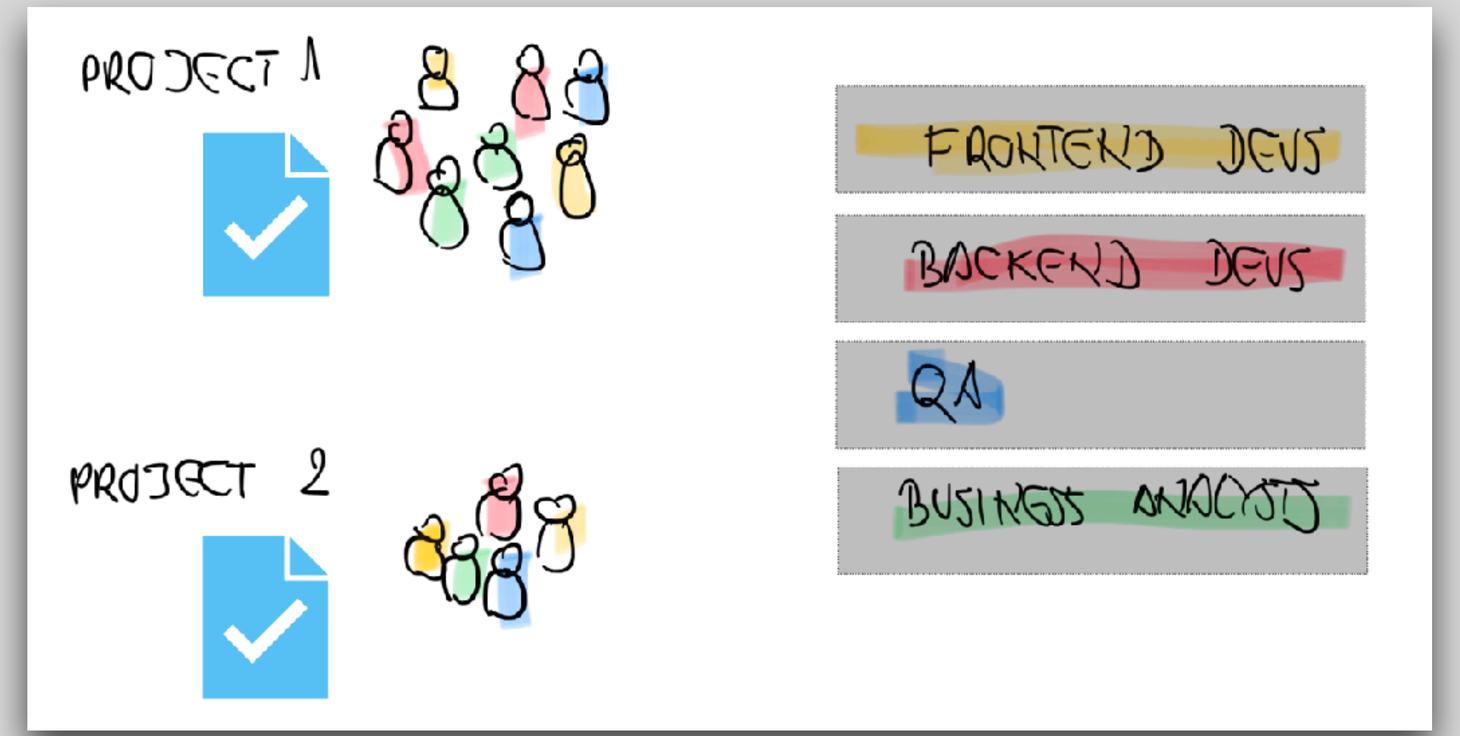


Low autonomy in solving customer problem

High autonomy in building

Slow delivery and a lot of dependencies

Resourced projects

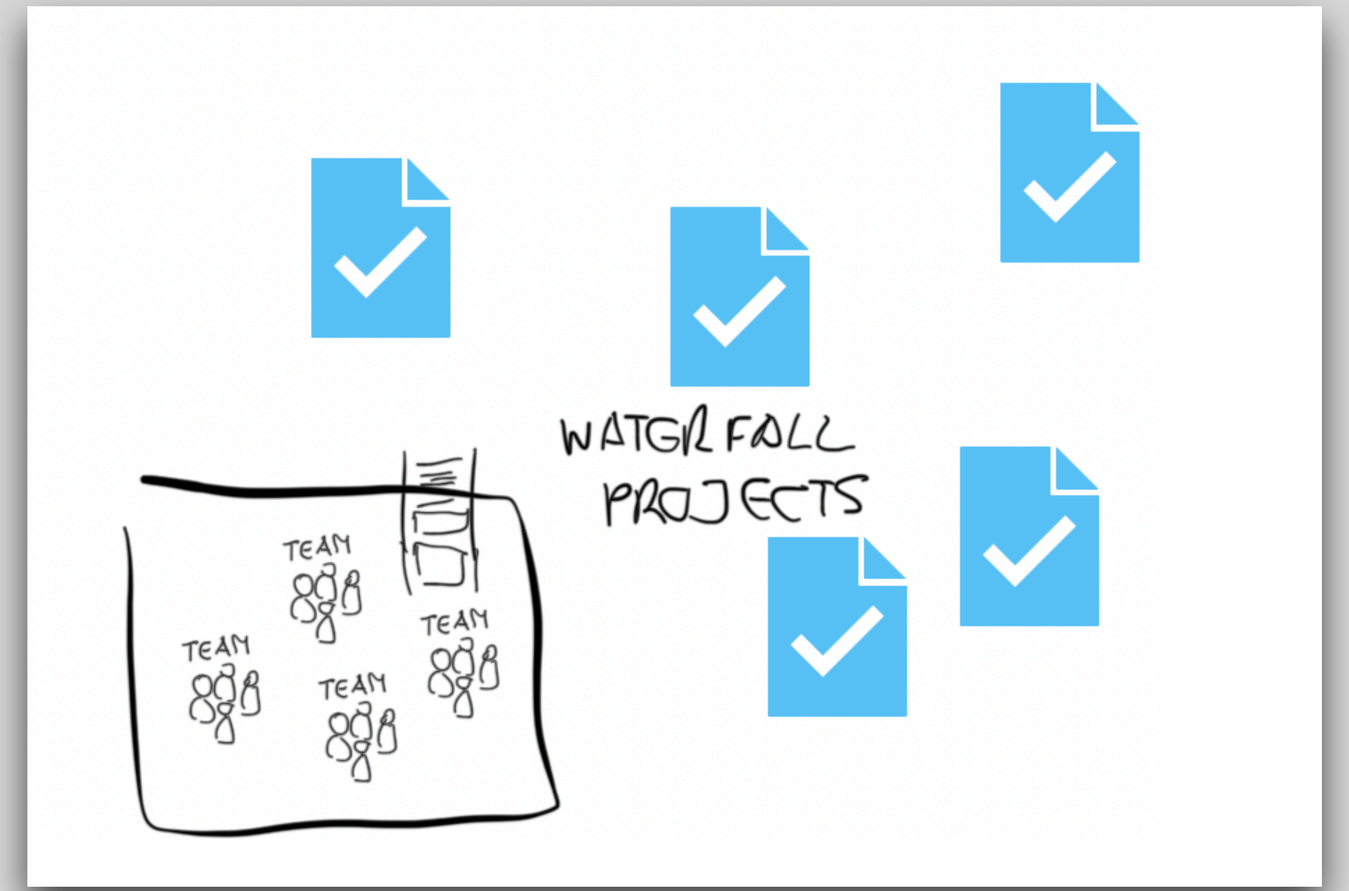


High autonomy in solving customer problem

Low autonomy in building

Maintenance hell

Isolated group

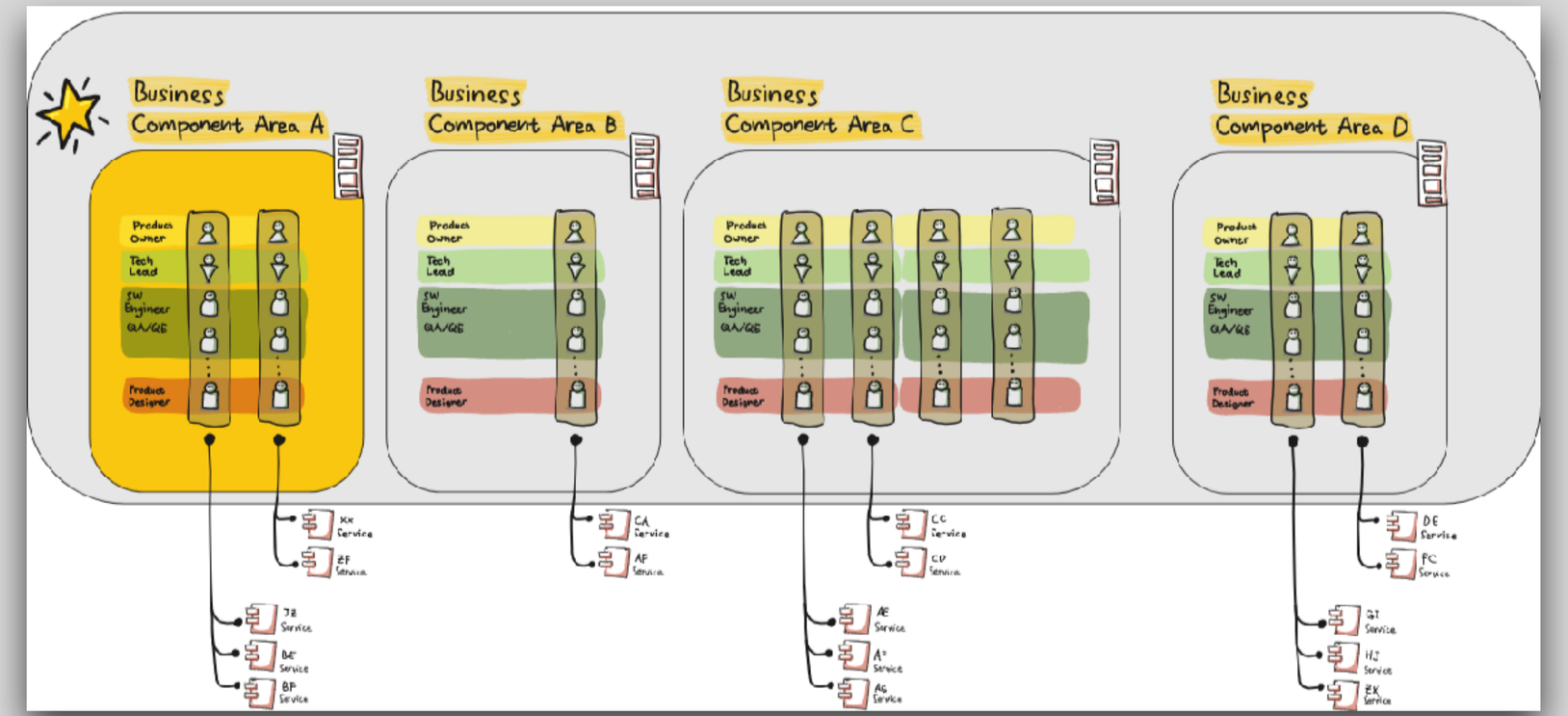


High autonomy in solving specific customer problems

High autonomy in building (except between other teams in the same group)

Dependencies hell with teams outside of group

Divide, give “autonomy” and ownership and then…

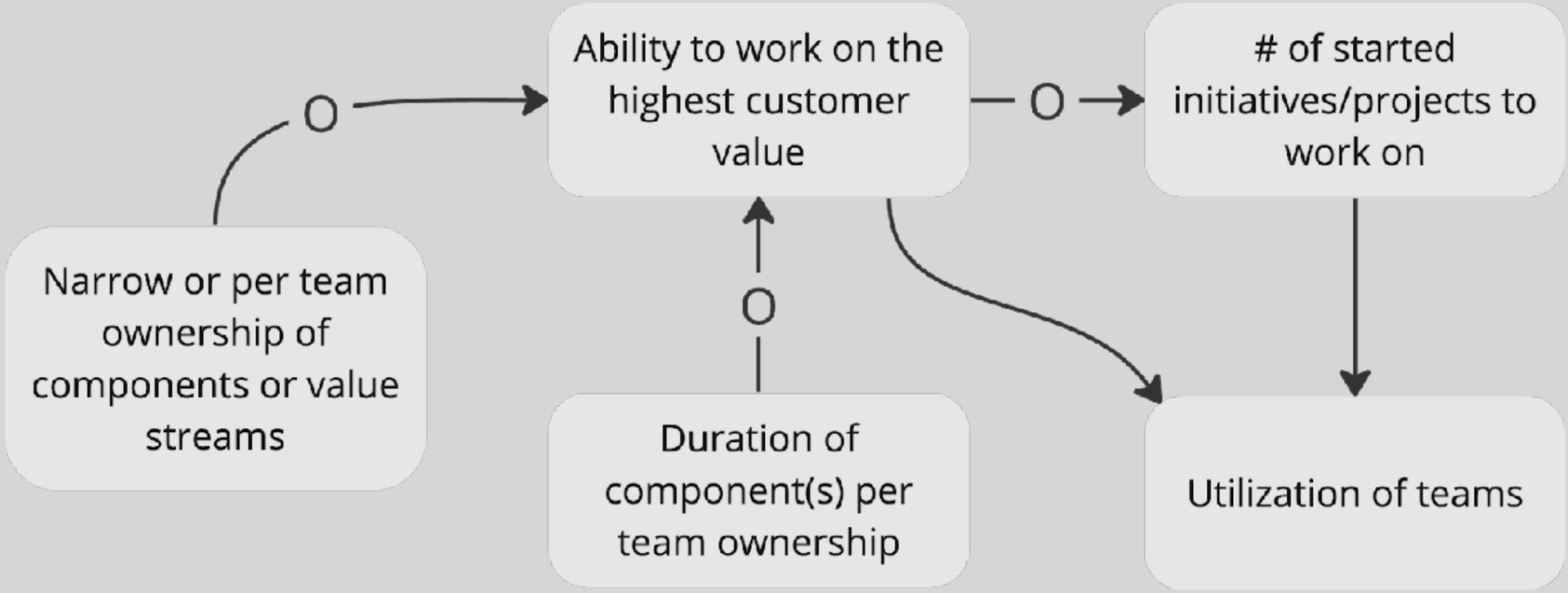


Low autonomy in solving customer problems

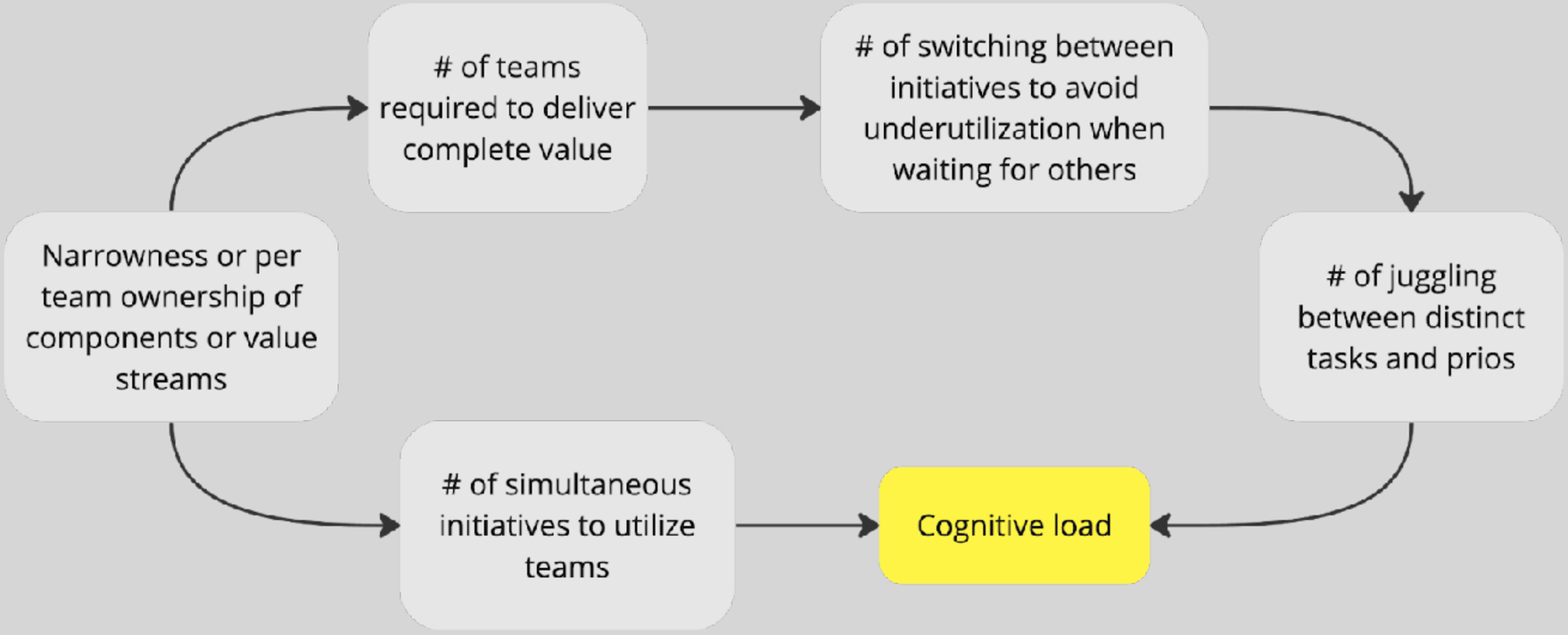
High autonomy in building

Slow delivery (unless done fully by a single team) and a lot of dependencies

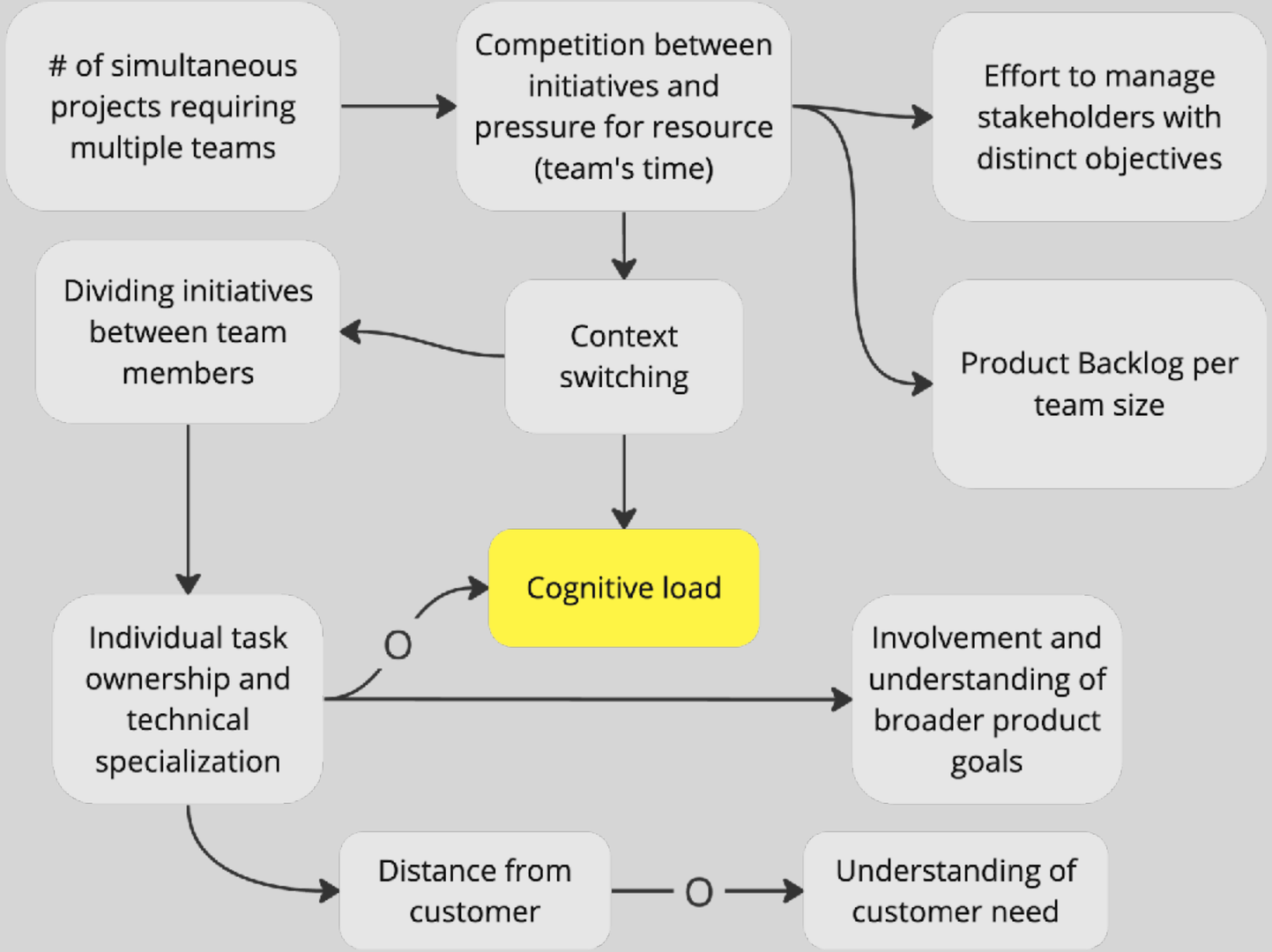
Component or value stream teams prevent prioritization



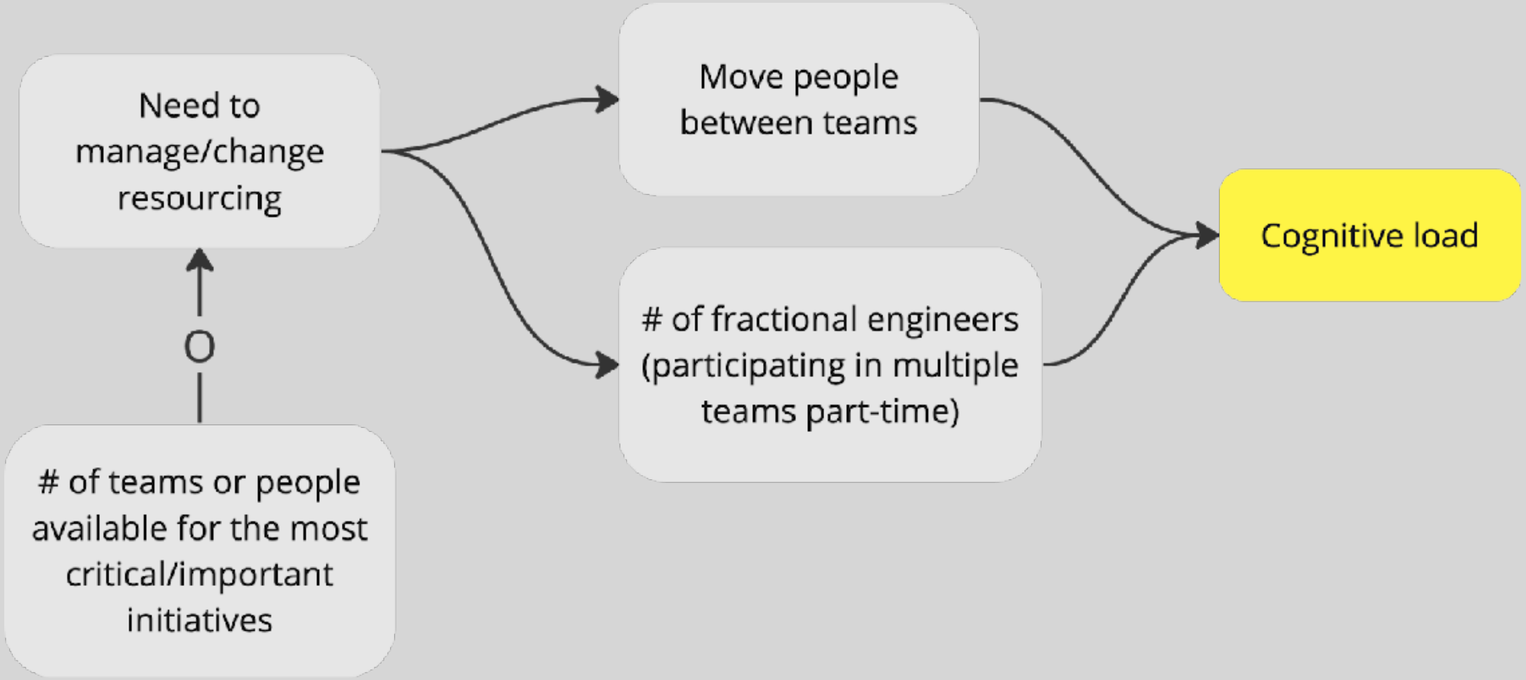
Limited scope leads to higher cognitive load



Waste of running many projects

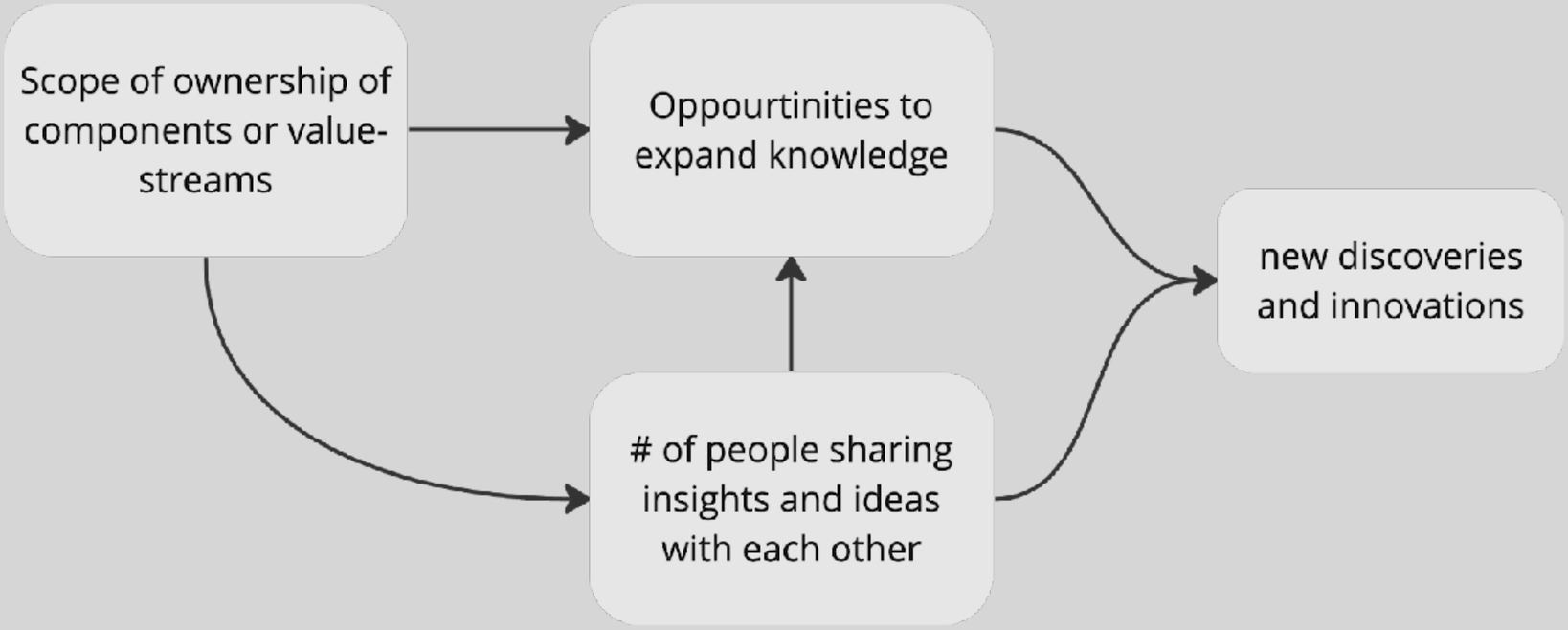


Change in prio leads to fractional engineers



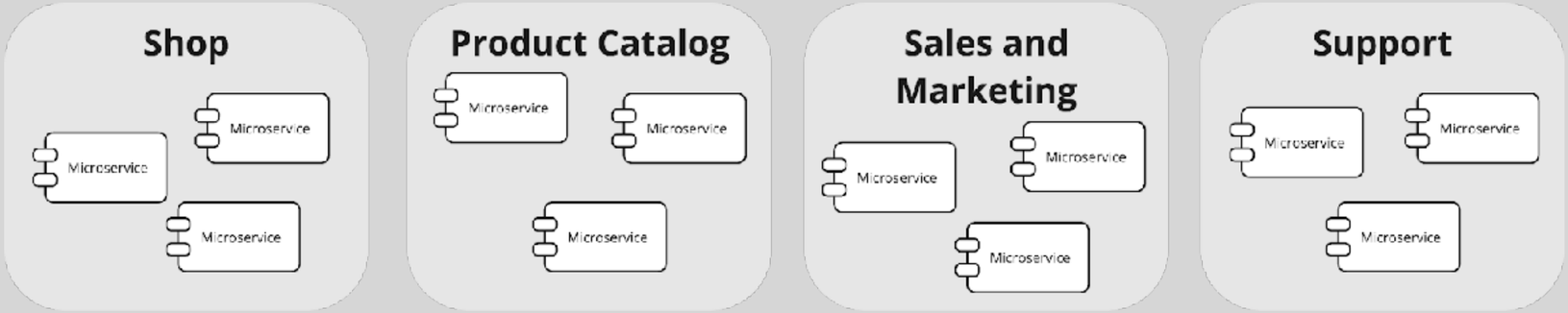
Narrow ownership leads to reduced focus

Larger scope of ownership leads to more innovation

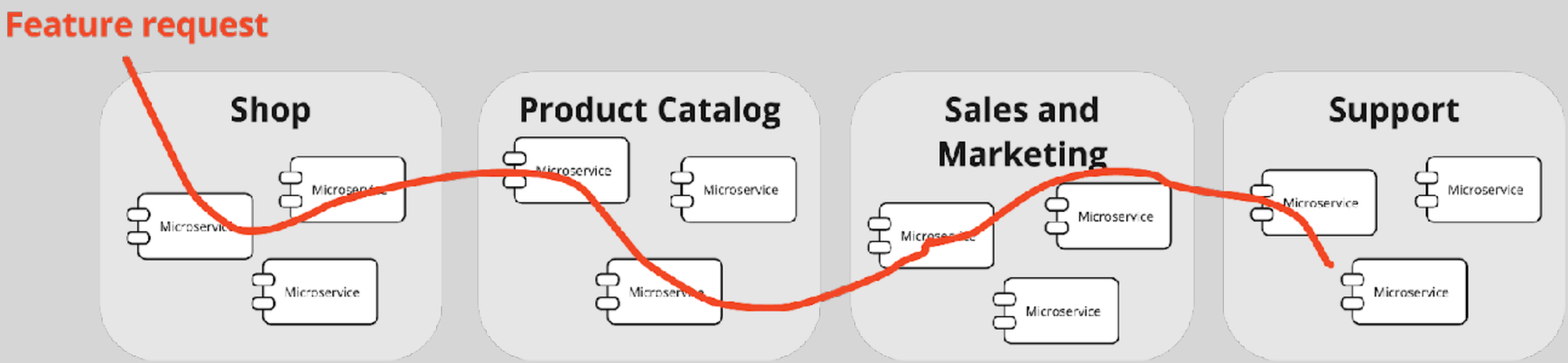


But Conway’s Law…

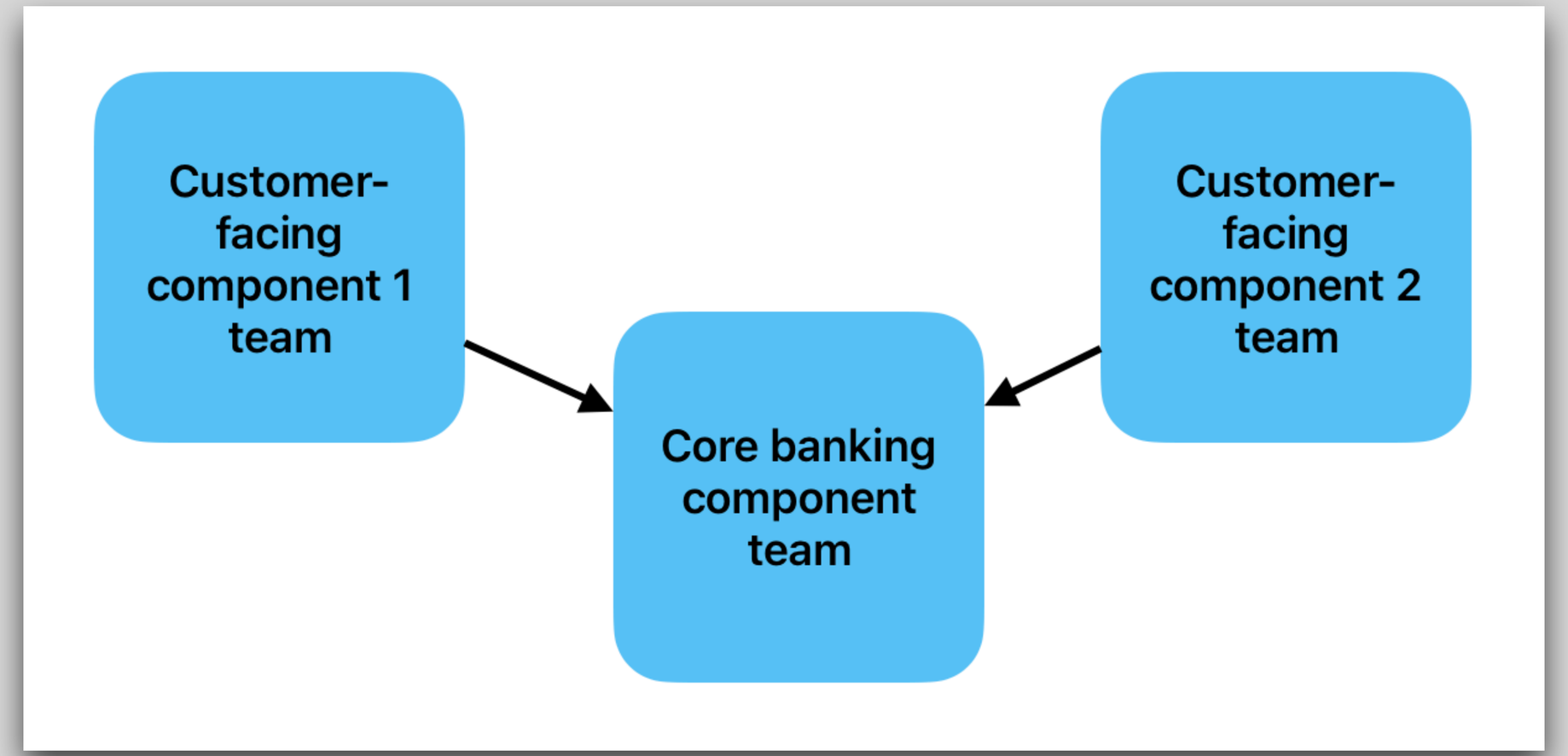
Team structure according to wanted architecture



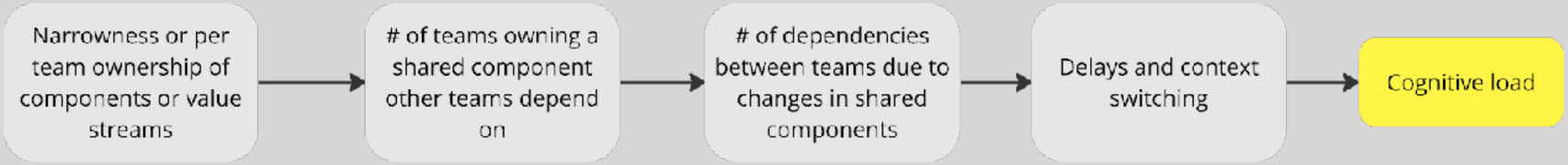
Customer need doesn't follow fixed structure



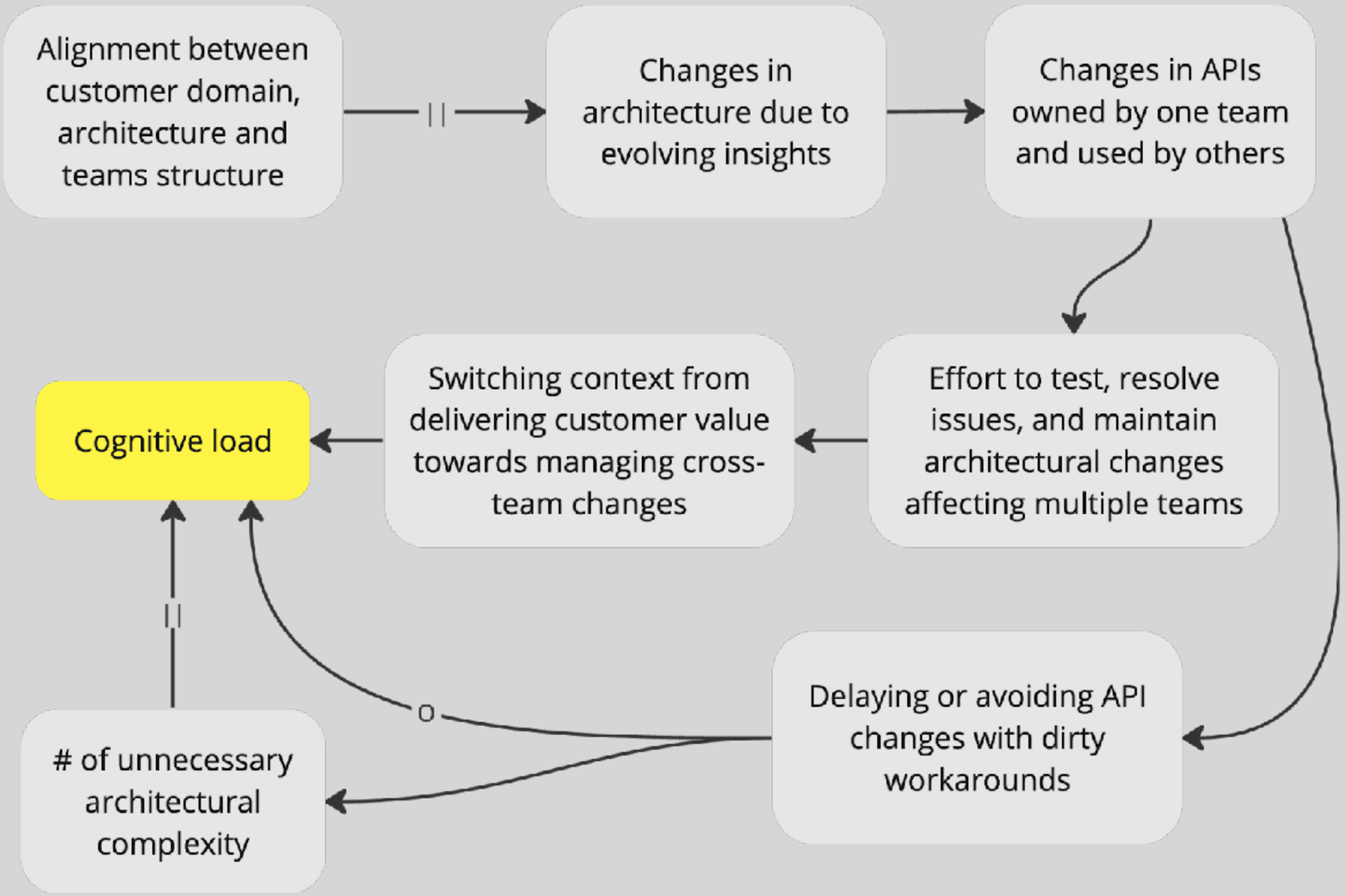
Shared components problem



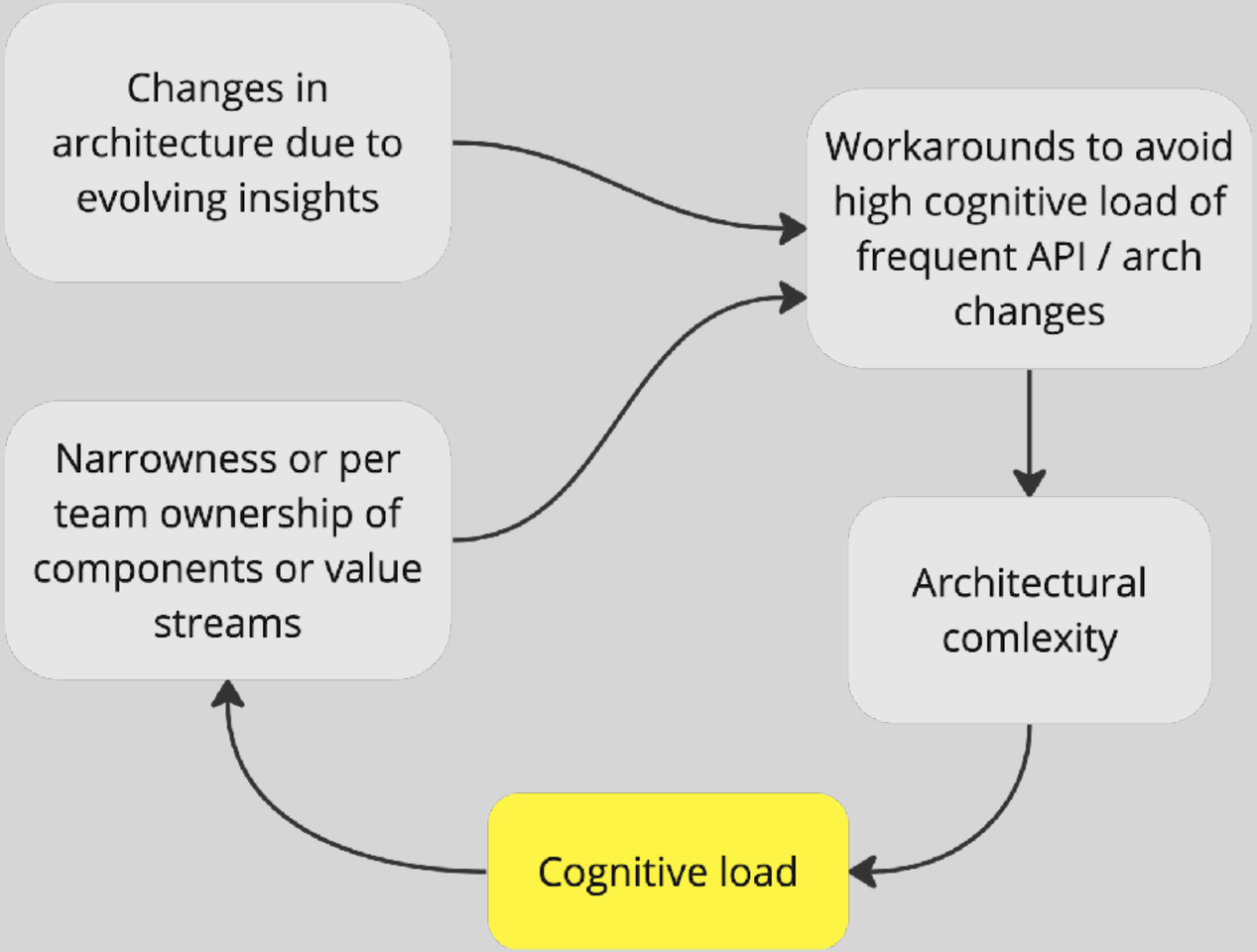
Dependencies lead to very high cognitive load



API changes become very expensive



Growing architectural mess

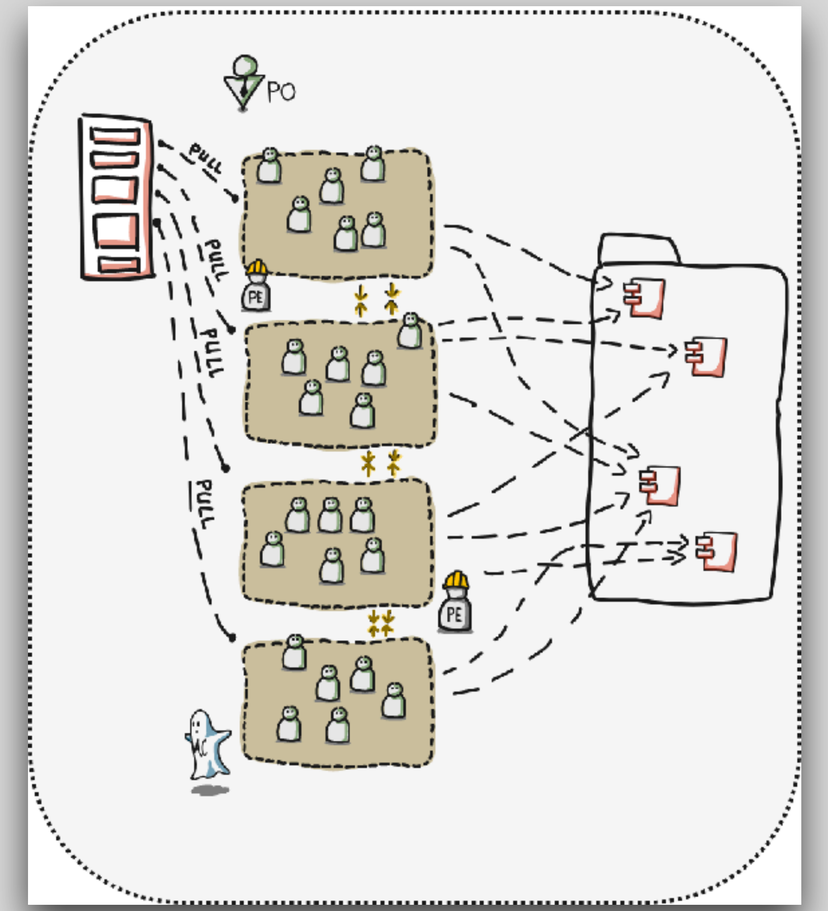


How to overcome Conway's Law?

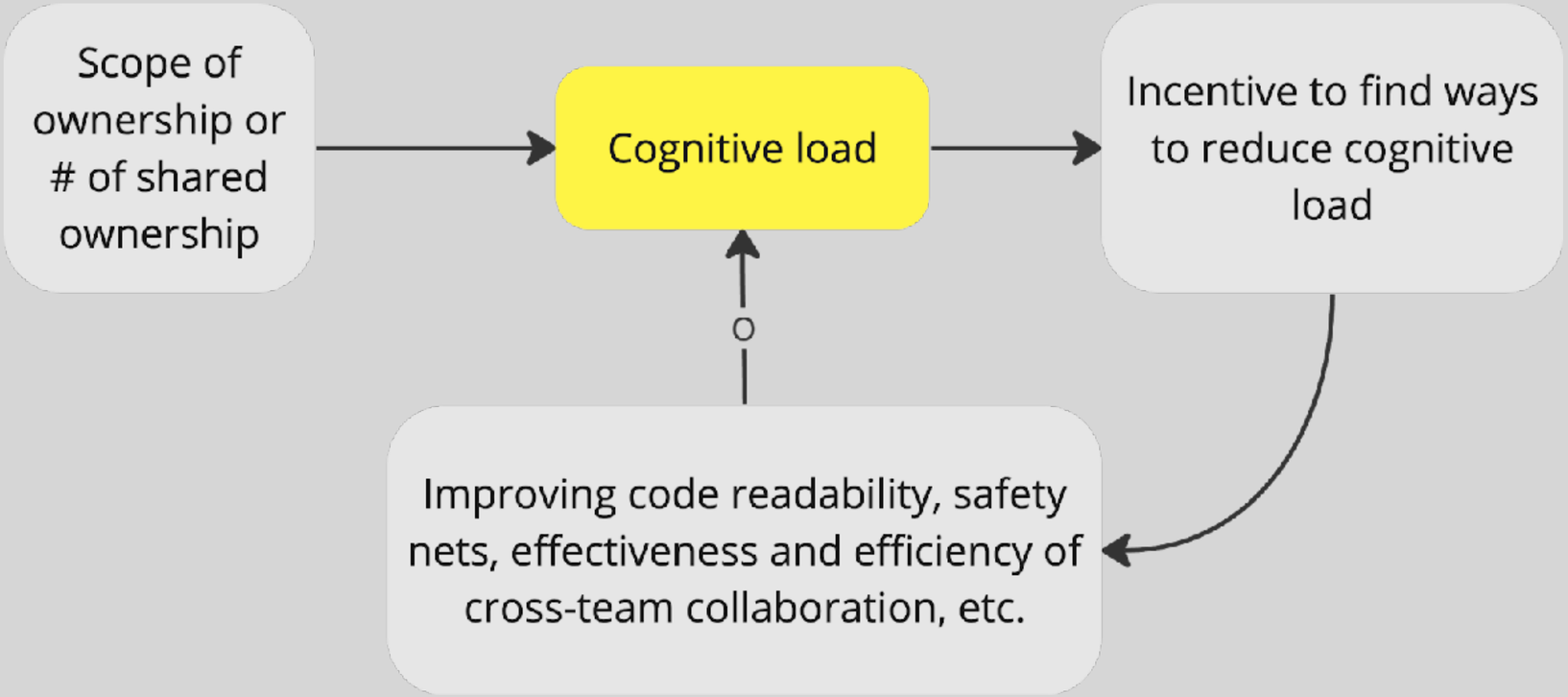
Each team works from whole product point of view

Each team CAN work on any component

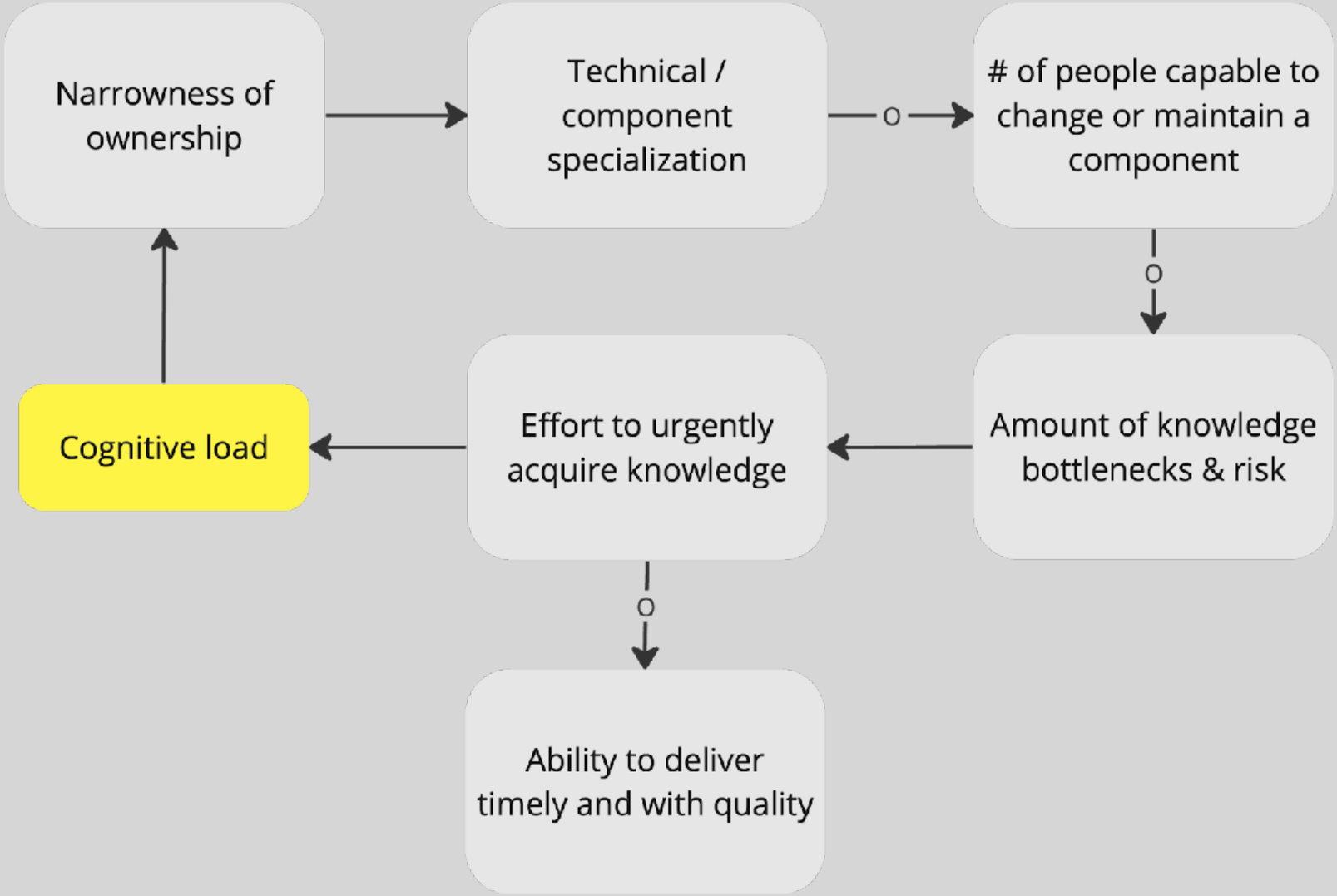
Each team works on the most important



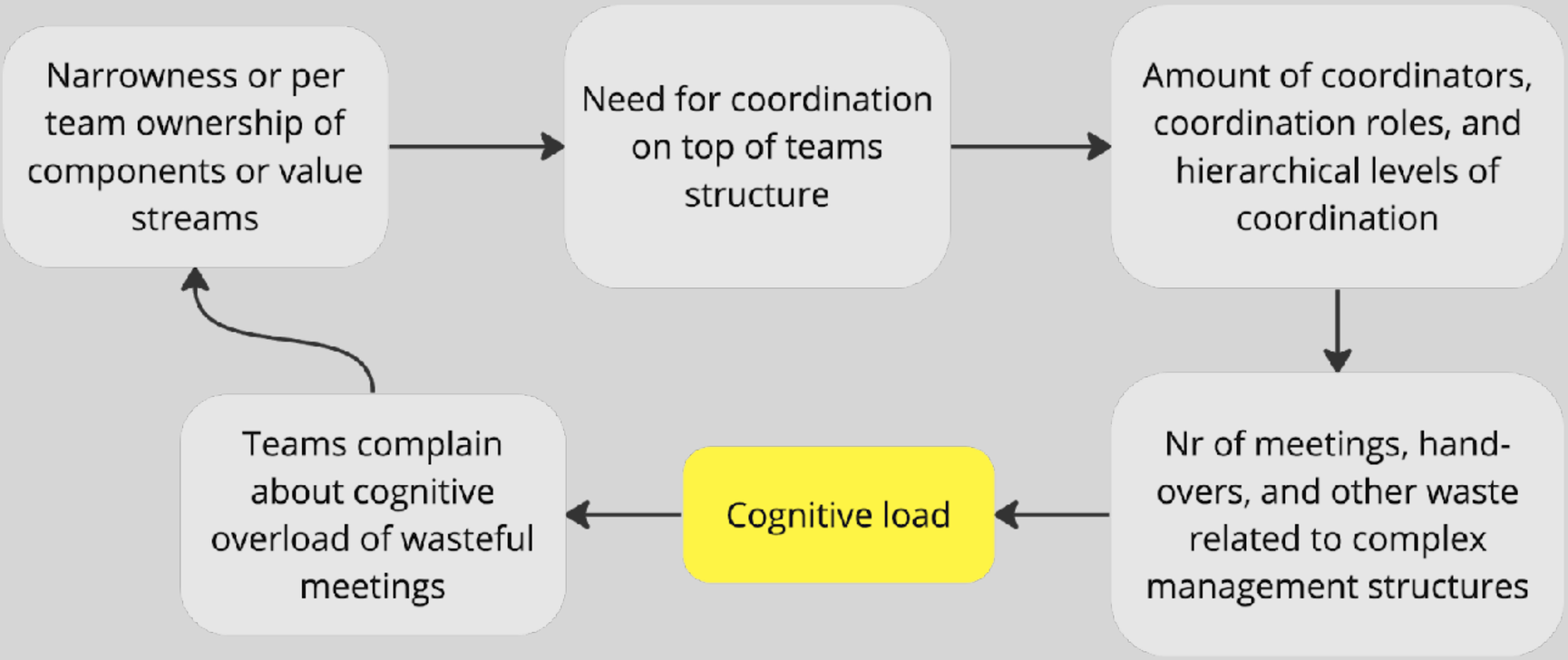
Isn’t shared ownership even worse?

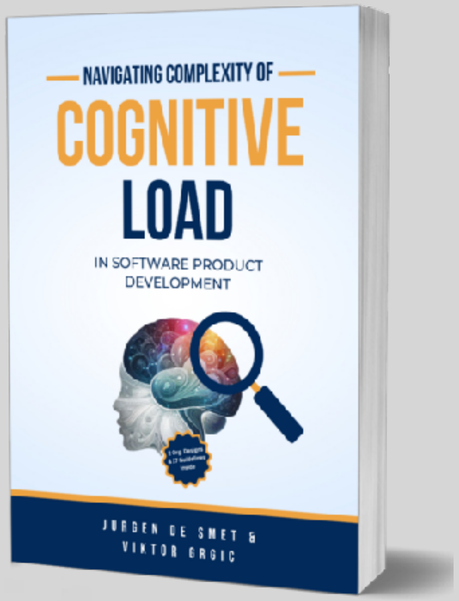


Isn’t shared ownership even worse?



“Too many meetings” problem



Scan for the FREE book

See: https://less.works/courses/less-courses