

開発スピードを  
落とさないために必要な  
イネーブルメント組織とは

Suguru Namura - Mar 8th, 2023  
CTO week 2023 by Offers

# 自己紹介 - Suguru Namura



Amebaブログ



AWA



入社

2004

2005

2006

2007

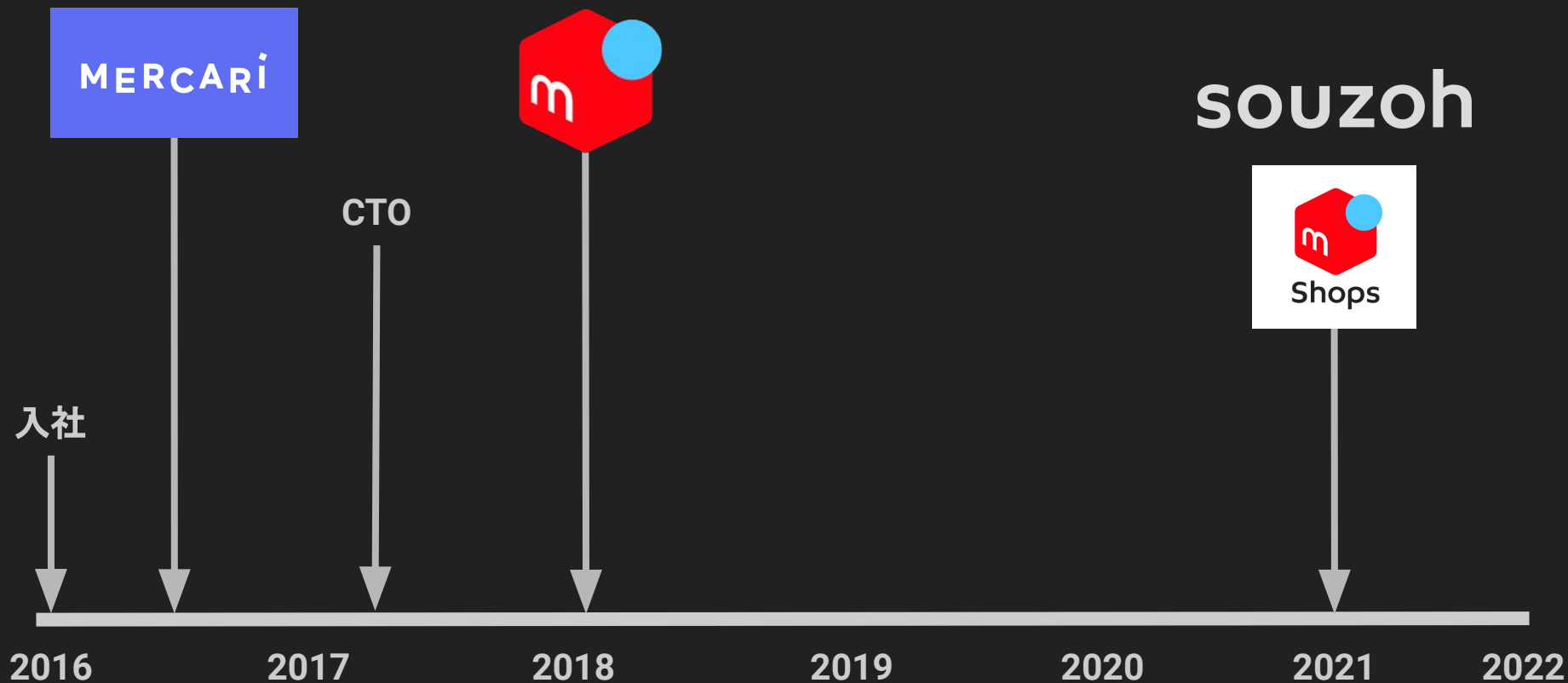
2009

2011

2013

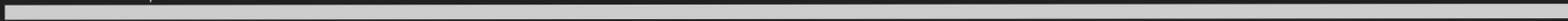
2015

2016





LayerX



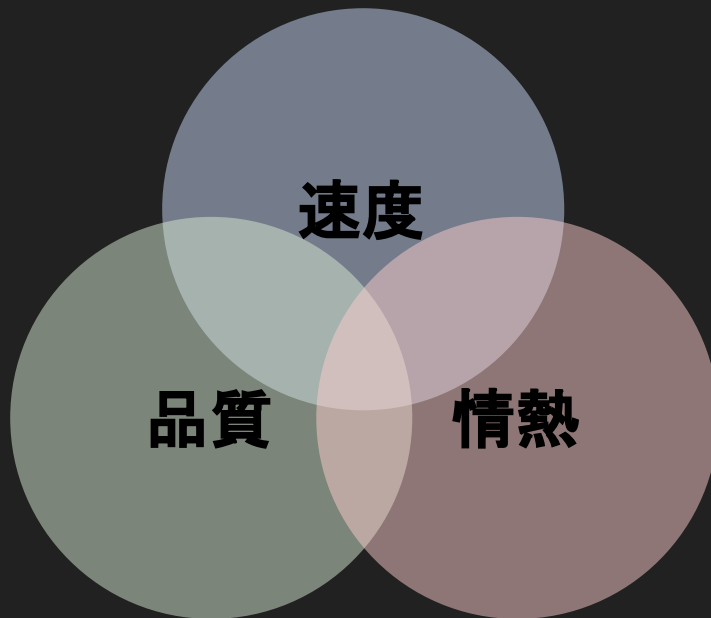
2022

# 今日話すこと

suguru が過去に経験したプロダクト開発を元に、  
単にプロダクトを立ち上げるだけでなく、  
プロダクトを持続的に成長させる上で重要な要素である  
**“Enablement”** について話します。

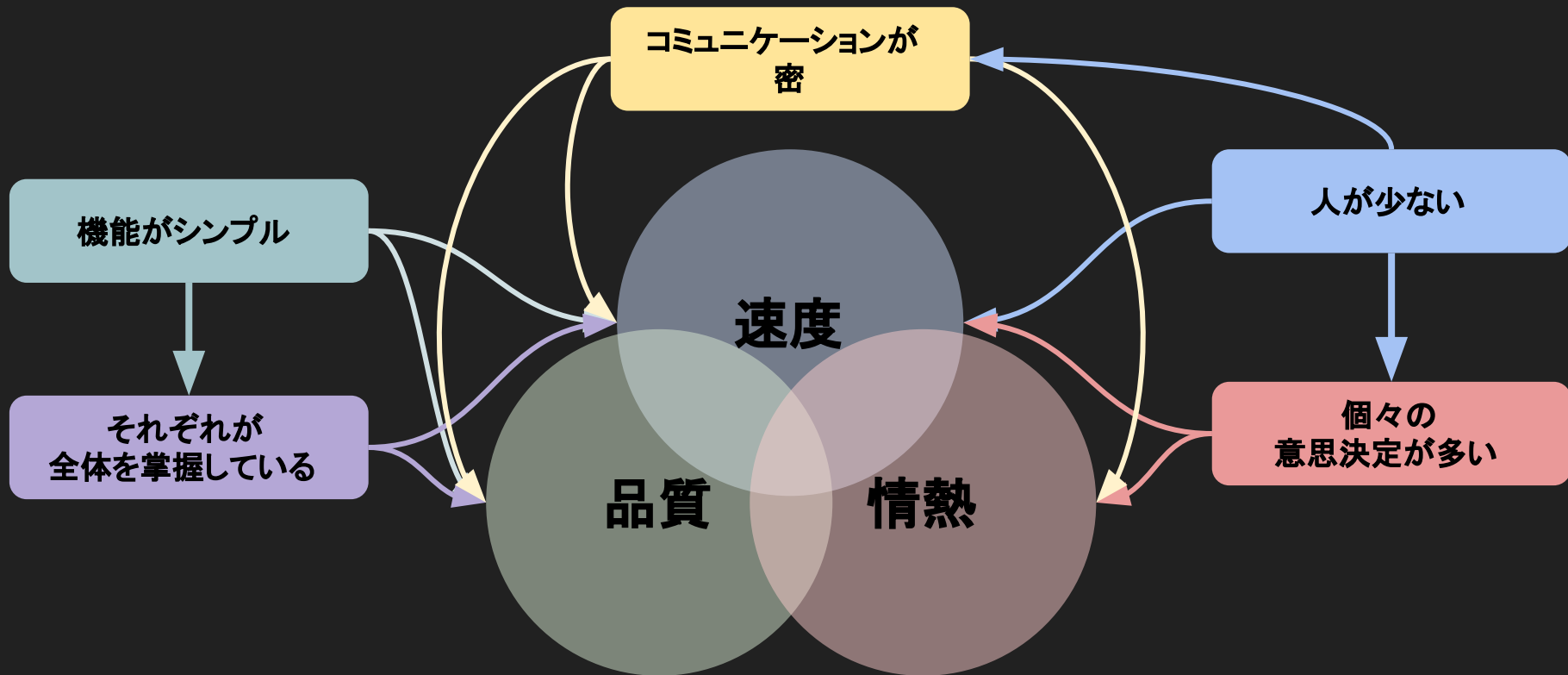
なぜプロダクトは成長と共に  
鈍化していくのか

# 持続的なプロダクト成長にとって重要な要素



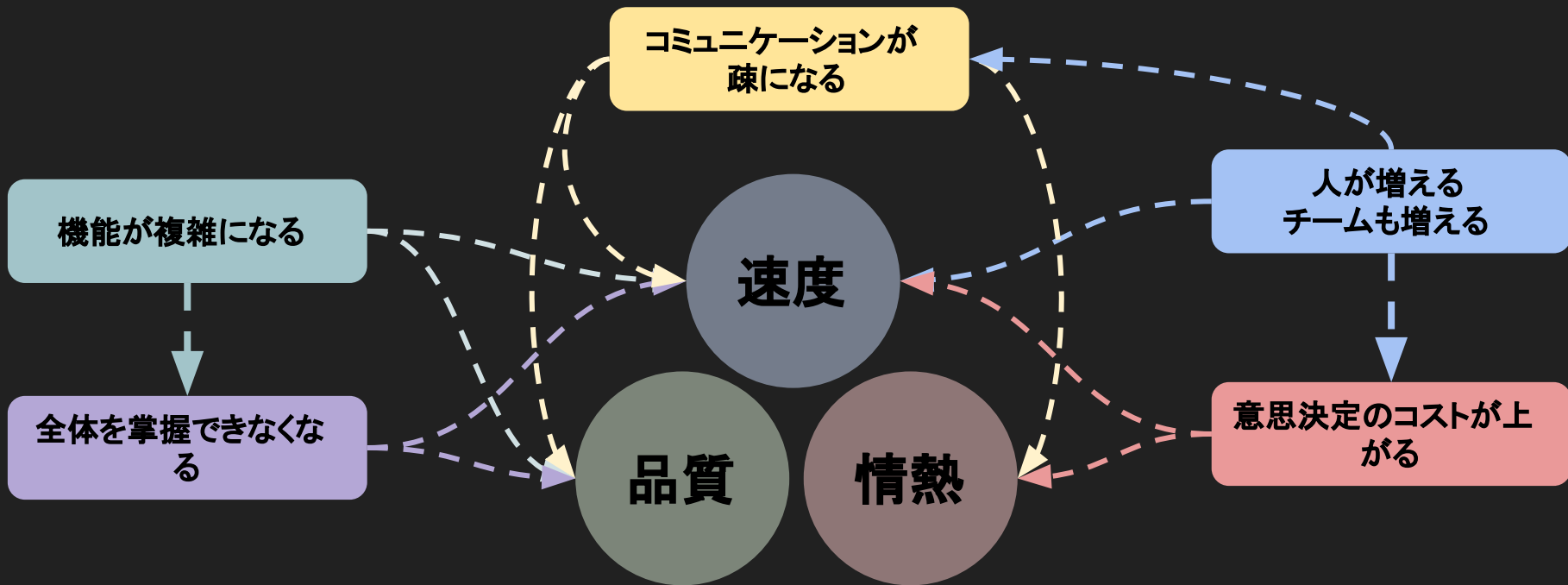


# 立ち上がり期は、どの要素も満たしやすい



プロダクトが成長すると ...

# すべての部分で負のインパクトが生まれる

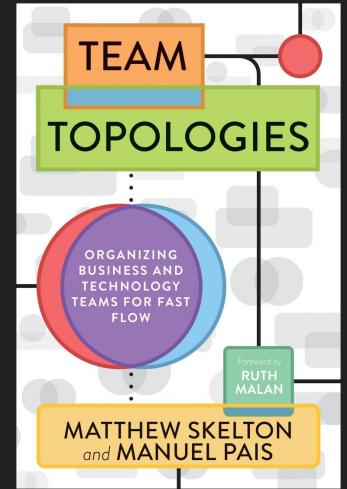


結果、速度・品質・情熱は連動してしぼんでいく ...

**プロダクトや組織が成長しても  
根幹となる要素が失われないために ...**

増えるチーム・人の  
パフォーマンスを最大化し続けることに  
挑戦する必要がある

# Lesson Learned from Team Topologies



<https://teampologies.com/>

# Team Cognitive Load - チームの認知負荷

ワーキングメモリにかける

心的負担の総量



頭をどれくらい使うか？

<https://teampologies.com/>

# Team Cognitive Load - 認知負荷

## Intrinsic

GraphQLのスキーマはどうやって定義するのか？

## Extraneous

アプリのデプロイをもう一度したいけど、どうすればいいか？

## Germane

銀行送金はどのように機能しているのか？

<https://videos.itrevolution.com/watch/527514898/>

<https://teamtologies.com/>



# Team Cognitive Load - 認知負荷

**Intrinsic**

スキル

**Extraneous**

仕組み

**Germane**

ドメインフォーカス

<https://videos.itrevolution.com/watch/527514898/>

<https://teampologies.com/>

## Team Cognitive Load - 認知負荷

プロダクトチームは、どこにワーキングメモリを使うべきか？

**Gemane**

ドメインフォーカス

# Lesson Learned From Team Topologies

ソフトウェアサービスやプロダクトの  
サイズを最小化し  
認知負荷を最小化する

<https://teampologies.com/>

# Lesson Learned From Team Topologies

Team First でサービス境界を作る

小さいチーム ~9人

サービスへの完全なオーナーシップ

## Team Topologies に登場する 2つのチーム

**Stream Aligned Team**

**Enabling Team**

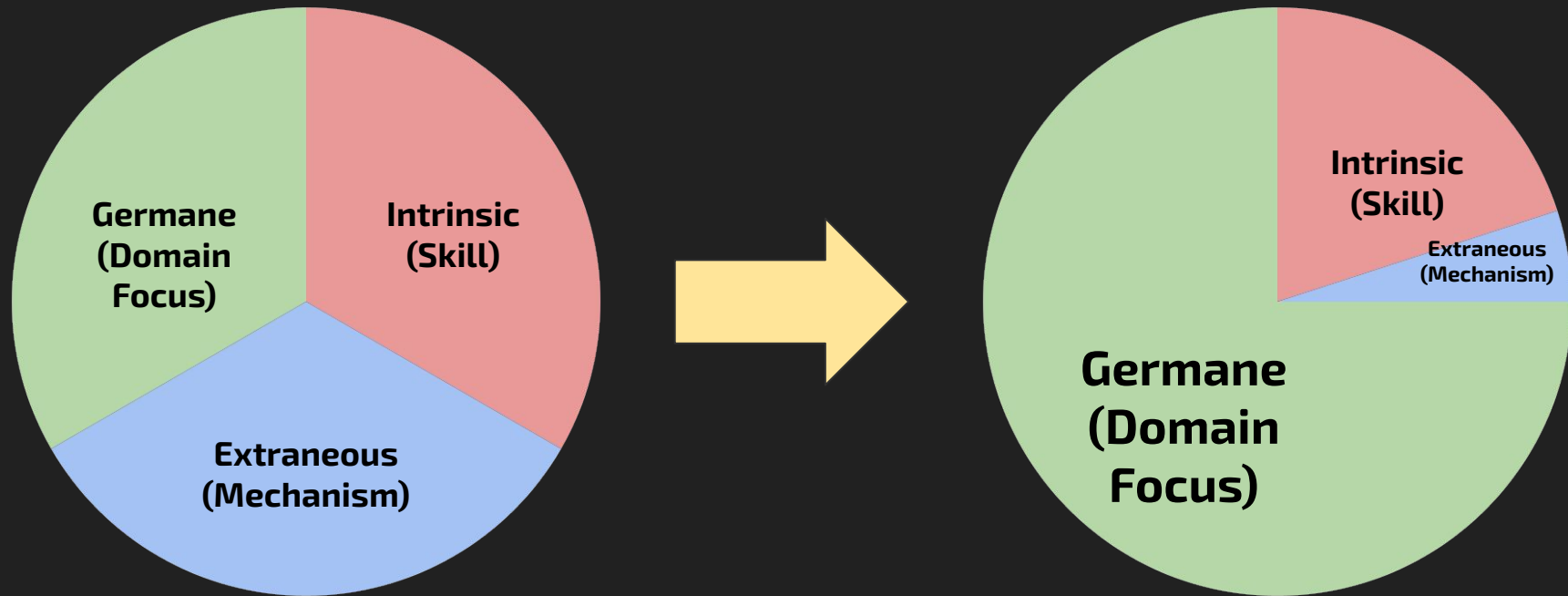
**重要なこと...**

**小さなチームが完全なオーナーシップを  
自らの意思決定で開発を進められること**



**Stream Aligned Team の  
パフォーマンスを最大化する**

# Enabling Team が目指すプロダクトチームの Stream-Aligned Team の 認知負荷バランス



# Enabling Team の目指す状態とは？

Stream Aligned Team がドメインフォーカスした領域に頭をフルコミットするために...

**新しい仕組みや技術の導入を共に実装する**

**Team Cognitive Load の軽減のための  
仕組みを整備する**

**ドメインフォーカスするための  
ツールを提供する**



# LayerXでの Enabling 事例

lxdev という開発用TUI

<https://tech.layerx.co.jp/entry/2022/12/12/131507>

**Extraneous**

```
Processes
-----
61867|Running|http://localhost:8090|develop|0.12%|81.14MB
61848|Running|http://localhost:8091|develop|0.12%|80.81MB
-----
60335|Running|http://localhost:3004|develop|0.00%|2.42MB
-----

id-api

Starting
10:52AM I
bel=serve
bb9edfe0ec5 la
[2022-12-
layerxcom
suguru/github/

(s)tart (q)uit (l)og (r)eload (e)dit (b)rowse

(s)tart (q)uit (l)og (r)eload (e)dit (b)rowse (tab)switch panel
```

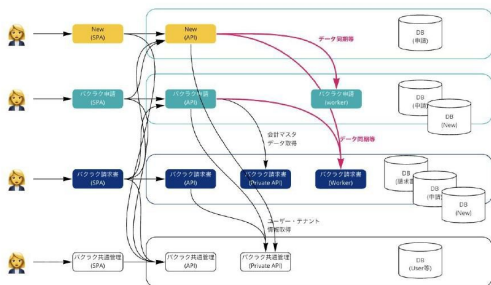
# LayerX での Enabling 事例

## Team Cognitive Loads を意識したアーキテクチャ

[https://levtech.jp/media/article/column/detail\\_166/](https://levtech.jp/media/article/column/detail_166/)

参考: 現状のアーキテクチャイメージ

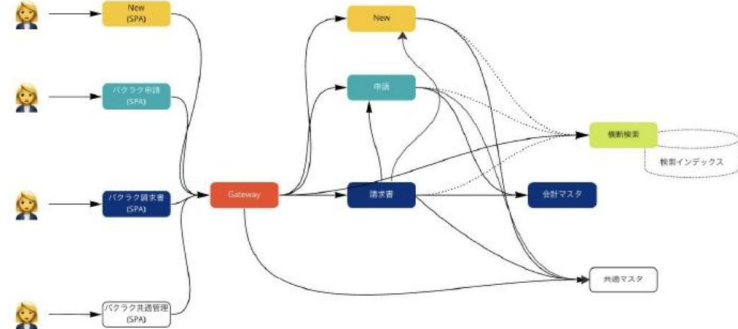
IIJ IIX



- 各プロダクトチームごとにコンポーネントがある、コンウェイの法則そのもの。
- 歴史的経緯で、フロントがばらばらの API を叩いてしまっている (Gateway が無い)
- 裏側で **データ同期** が必要になっている (一部のリソースのみ)
  - 各種要件上仕方がなかった
  - 連携するプロダクトが増えるたびに地獄になっていく
- 循環参照を鬼のように気を遣って避けているが、深いドメイン知識と未来予知スキルが必要



After



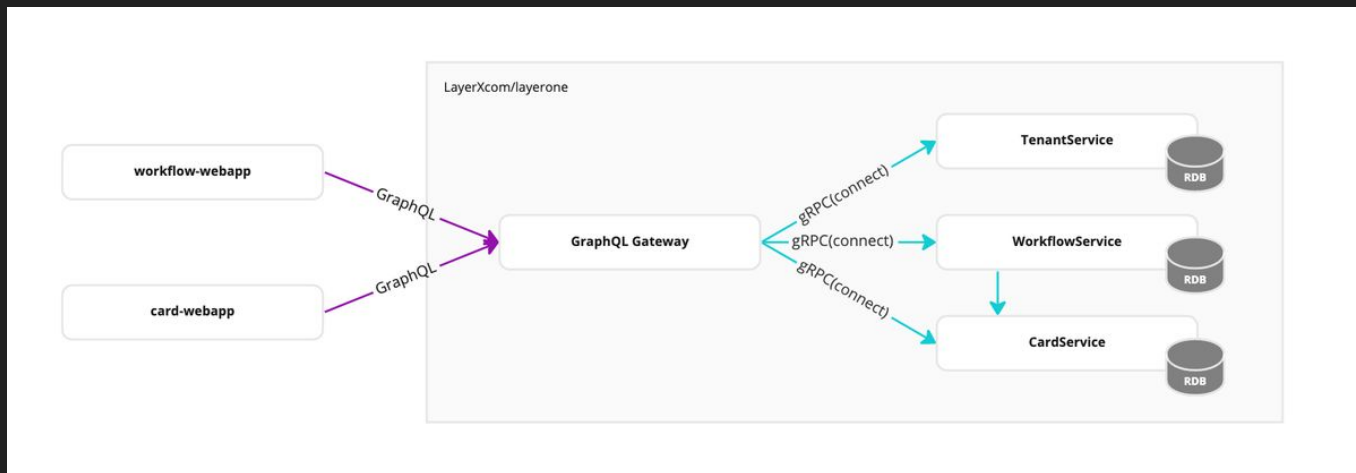
Intrinsic

Extraneous

# LayerXでの Enabling 事例

REST API on GraphQL

<https://tech.layerx.co.jp/entry/2022/12/19/230000>



**Intrinsic**

**Extraneous**

# LayerXでの Enabling 事例

## ChatGPT on Slack

<https://tech.layerx.co.jp/entry/2023/03/06/chatgpt-on-slack-new-platform>



**suguru** 1:50 AM

@ChatGPT 誕生おめでとう、最初に何をしたいですか？



**ChatGPT** APP 1:50 AM

私は人工知能であり、祝福を受け取ることができますが、人のような願いや欲求を持っていません。私はプログラムされたタスクを実行することができますが、自己意識や欲求を持っていません。あなたが私に何を望むかに応じて行動します。

**Extraneous**

**Fun!**

つまり Enabling Team は...

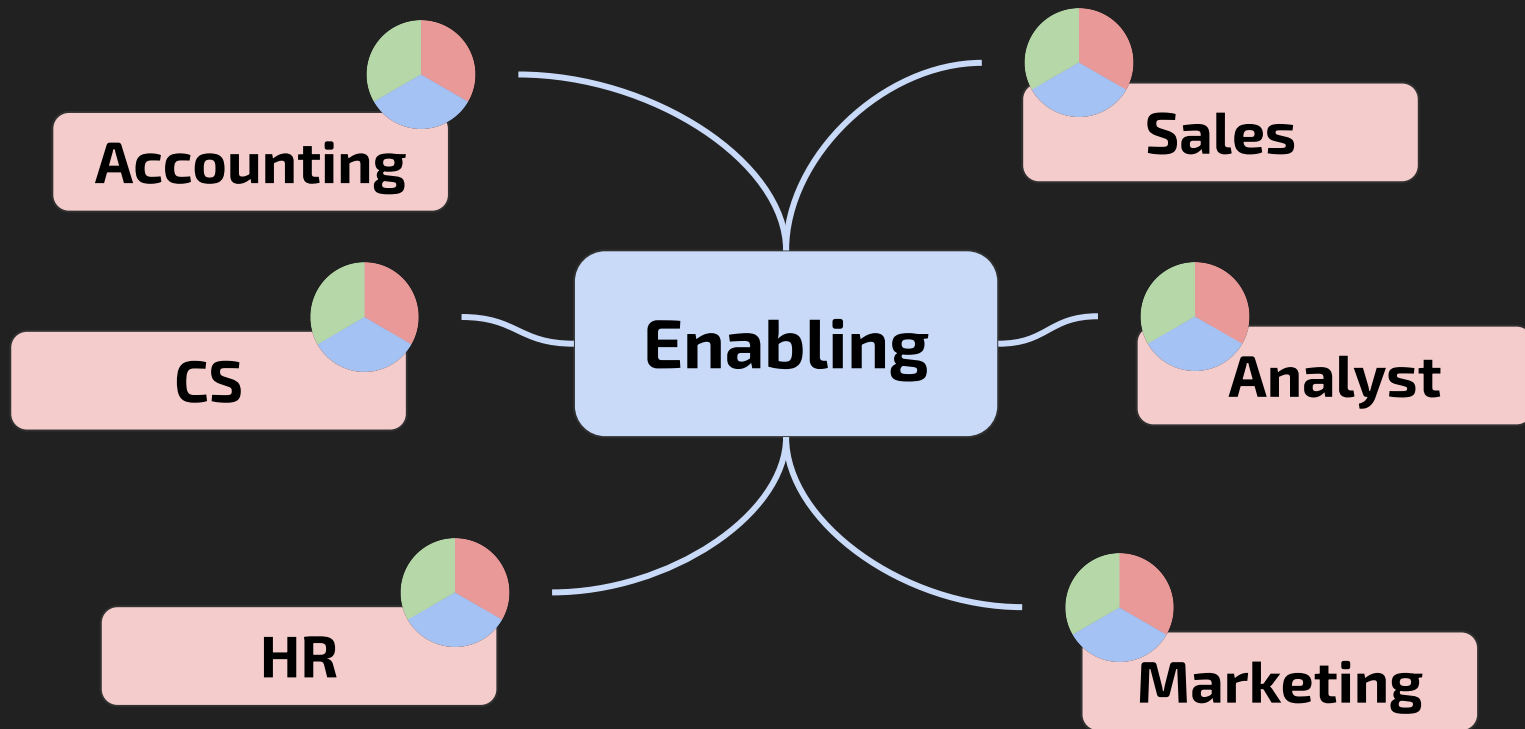
成長するプロダクトにおいて

プロダクトの開発に必要な認知負荷を下げ

パフォーマンスが最大化する「小さなチーム」が

ドメインにフォーカスする状態を目指す

# Enablement はプロダクト組織のためだけではない



**Enablement の概念を取り入れ  
会社全体の速度を上げることができる！**