Chapter1　組織図の問題

そもそもTeam Topologyとは

- ソフトウェアを顧客にデリバリするための組織の考え方について書かれた本
- 書内のスコープだとソフトウェアを作る組織をターゲットにしているが、拡張して組織運営の文脈で参照されることが多い
- チートポ文脈という謎の言葉が存在するほど

要点

- コンウェイの法則では、ソフトウェアアーキテクチャーとチームインタラクションを同時に設計する利点を説いている。両者に働く力は同じものだからだ
- チームトポロジーはチームの目的と責任を明確にし、チーム間の相互関係の効果を向上させる
- チームトポロジーでは、戦略適応性の実現のために組織を調整しつつ、ソフトウェアシステムにおいては人間的なアプローチを利用する

背景

- 企業が最適化の目的を安定性と速度で選べる時代は終わった
  - ソフトウェアは安全に提供運用する必要がある
  - 企業として成長する必要もある
  - 古くからの組織体制だと実現できない

これをやればいい、というものでもない

- 正しいツールとプロセスを使っていればチームが成功するという訳ではない
  - 状況に応じて動的に組織構造は変化させる必要がある

つまりチームトポロジーとは？

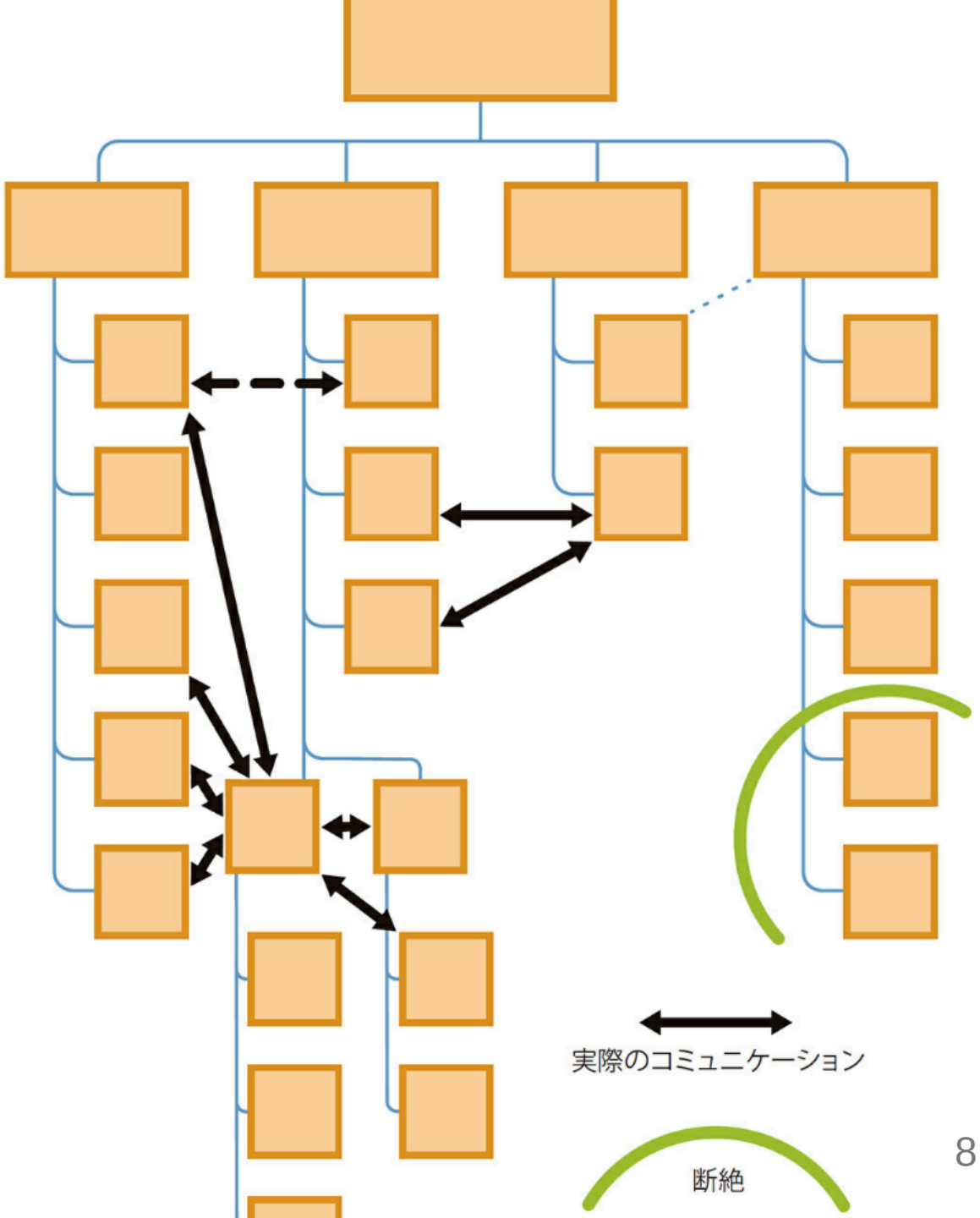
- ビジネスの速度と安定性を実現する技術組織の適応型設計モデル
  - つまり考え方に近い
- いろんな環境変化に対応するためのもの

組織のコミュニケーション構造

階層型の組織図をそのままコミュニケーションの文脈に当てはめようとすると失敗する

階層型（従来型）

- よくある組織図のやつ
- レポートラインがツリー型の上下に設定されている



階層型のPros/Cons

Pros

- 規制統制やコンプラ遵守には適している
  - だめだめな訳では無い

Cons

- レポートラインの考えはコラボレーションには適さない
  - 仕事を進めるうえで横のチームと連携するのはよくあること
- アウトカムが安定しない
- 局所最適化の温床になる

ついでに

- アウトプット
  - 自分が出力したもの
- アウトカム
  - 成果、効果
  - アウトプットを受けて、顧客がどう便益を得たか

参考、引用) https://note.com/chan_niwa/n/n50380f931dac

じゃあどうすべきか？

- 全体の最適化に焦点を置く
  - ワークフローの現時点でのボトルネックを解消する。を回す。
  - チームが新しい状況に素早く対応し、価値のデリバリーを高速かつ安全に実施することに焦点を当てる
    - そのためにはチーム構造もインタラクションの方法も動的に変化して実現させる
- 組織図もアーキテクチャ図と同様にすぐに陳腐化する

組織図からの脱却

そもそも組織構造には三つに分類される

1. 公式な構造：コンプライアンス遵守を円滑にする
2. 非公式な構造：個人間の「影響力の領域」
3. 価値創造構造：個人間やチーム間の能力に基づいて、実際にどう仕事を終わらせるか

組織構造としての役割が違う

1. 公式な構造：コンプライアンス遵守を円滑にする

   →階層型組織図が適している

2. 非公式な構造：個人間の「影響力の領域」

   →価値創造に必要

3. 価値創造構造：個人間やチーム間の能力に基づいて、実際にどう仕事を終わらせるか

   →価値創造に必要

非公式な構造と価値創造構造が重要

従来の組織設計に欠けていたものは？

- 静的な設計なので、現代では効果的なアウトカムを産めない
- チームの成長とチーム間のインタラクションの考慮が必要

ついでに

- 組織設計の経験
    i. やむを得ない理由があるときに設計する
    ii. 設計判断のために選択肢を用意する
    iii. 適切な時期に設計する
    iv. 整合性が取れていない箇所を探す
    v. 将来を見据える
- むやみに組織を再編するとビジネスを破壊する

チームトポロジー：チームについての新しい考え方

- 以下四つに分かれる
  - ストリームアラインドチーム
  - プラットフォームチーム
  - イネイブリングチーム
  - コンプリケイテット•サブシステムチーム

チームトポロジー：チームについての新しい考え方

- インタラクションは以下三つにわかれる
  - コラボレーション
  - X-as-a-Service
  - ファシリテーション

内容は後章にて言及

つまり何なのか？

- 戦略変更が必要になったタイミングを感知する
- そのタイミングでチームがソフトウェアを簡単に構築、実行する
- オーナーシップも持つ
- 人間的なアプローチを重視
    - 認知容量に限界がある事を前提にする
    - コンウェイの法則を大きな土台にしている

コンウェイの法則

「システムを設計する組織は、その構造をそっくりまねた構造の設計を生み出してしまう」

- (チームの認知容量を越えるような)モノリスアーキテクチャは分解しないといけない

たとえば

エリック・レイモンドは著書『ハッカーズ大辞典 改訂新版』（アスキー、2002年）のなかで、ユーモア混じりにこう述べている。

「コンパイラを作るのに４つのグループが作業していたら、できあがるのは４パスコンパイラになる」

逆に言うと

- システムに反映したいアーキテクチャーに合うようなチーム構造にする
- (チームの認知容量を越えるような)モノリスアーキテクチャは分解しないといけない

認知負荷とボトルネック

- 個人と同じくチームにも認知容量はある
- チームの責任やソフトウェアの担当範囲を決めるときに、認知容量は意識されない
    - これはよくない

内発的動機づけへの影響

- 認知負荷/コンテキストスイッチが増えすぎると、内発的動機付けが失われる
  - 自立
  - 熟練
  - 目的

つまり

- チームの認知負荷を制限する
- チーム間コミュニケーションも明確に設計する

必要がある

まとめ：チーム構造、目的、インタラクションを再考する

- ユーザからのフィードバックを元に軌道修正することに意味がある
- DevOpsとかそこら辺のおかげでテレメトリツールは発展している

まとめ：チーム構造、目的、インタラクションを再考する

- ダメな事
  - 組織図を弄る
- やるべき事
  - コンウェイの法則に納得する
  - 認知負荷の制限
  - チームファーストのアプローチ
  - インタラクションの促進

チームトポロジーのゴール

- コラボレーションの実行に集中
- オーバーヘッドを、組織が適応しながら動的に見つけられる仕組み作り

要点（再掲）

- コンウェイの法則では、ソフトウェアアーキテクチャーとチームインタラクションを同時に設計する利点を説いている。両者に働く力は同じものだからだ
- チームトポロジーはチームの目的と責任を明確にし、チーム間の相互関係の効果を向上させる
- チームトポロジーでは、戦略適応性の実現のために組織を調整しつつ、ソフトウェアシステムにおいては人間的なアプローチを利用する