DMMを支える決済基盤
技術的負債にどう立ち向かうか

Developers Summit 2025 Summer

合同会社DMM.com 藤井善隆

2025/07/17

決済基盤の状態

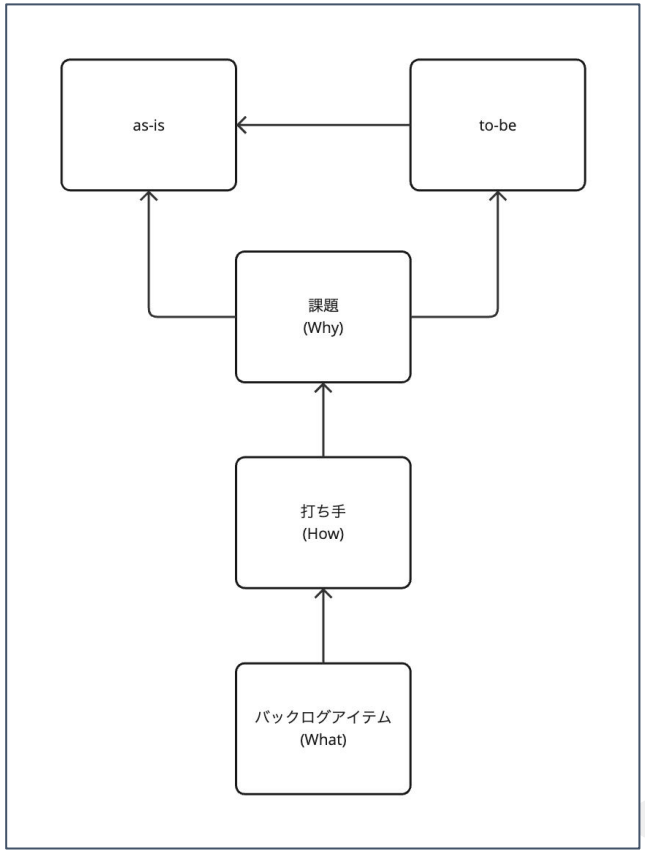
1. 多様なサービスの決済を社内向けプラットフォームとして構築
2. 複数の支払い手段がある
3. 2013年から現在のコードベース
4. DMMシステム全体は Microservices Architecture で構成している
5. 決済基盤内も複数のシステムで構成している
6. 各システムは異なる言語（PHP, Java, Kotlin, Go）
7. 各システムは異なる実行環境（オンプレ, EC2, ECS, EKS）
8. 各システムは異なるCI/CD（自家製, Jenkins, CircleCI, Github Actions, ArgoCD）

ミッション

- 決済基盤の技術的負債の返済に貢献すること
- 短期的ではなく、3-5年の中長期を視野に入れる

取り組み方

- **as-is** と **to-be** を得る
- 差分から **課題** を抽出
- **打ち手** を仮説
- **バックログアイテム** を設定
- バックログを消化するためのプロセスと体制を組織化

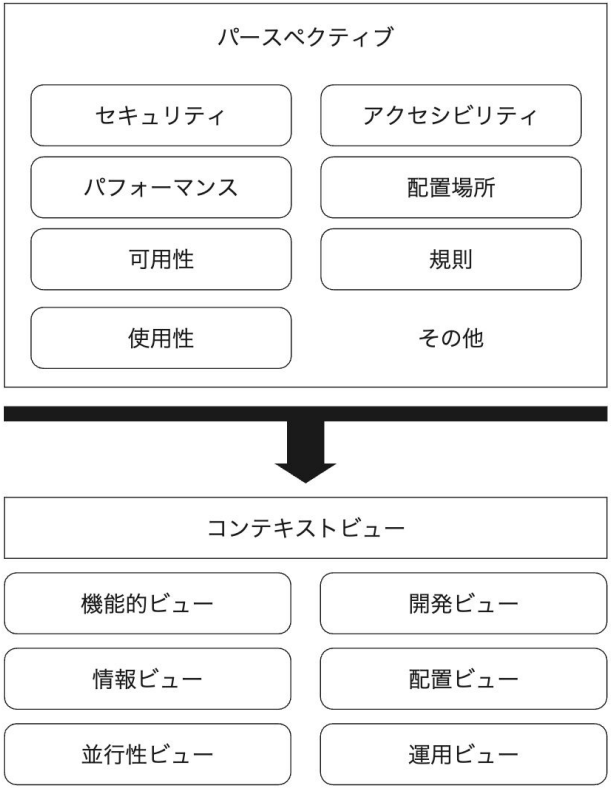


as-is の獲得

- **決済基盤** という システム
- システム の アーキテクチャ に注目する
- アーキテクチャ を見ていく観点漏れを防ぐ
- ビューポイント と パースペクティブ を活用する

ビューポイント と パースペクティブ

- 7つのビューポイントから
  アーキテクチャに着目
- パースペクティブは、
  ISO/IEC 25000 SQuaRE規格
  などを用いる
- 多面的かつ多角的に関心事を
  漏らさず分析する

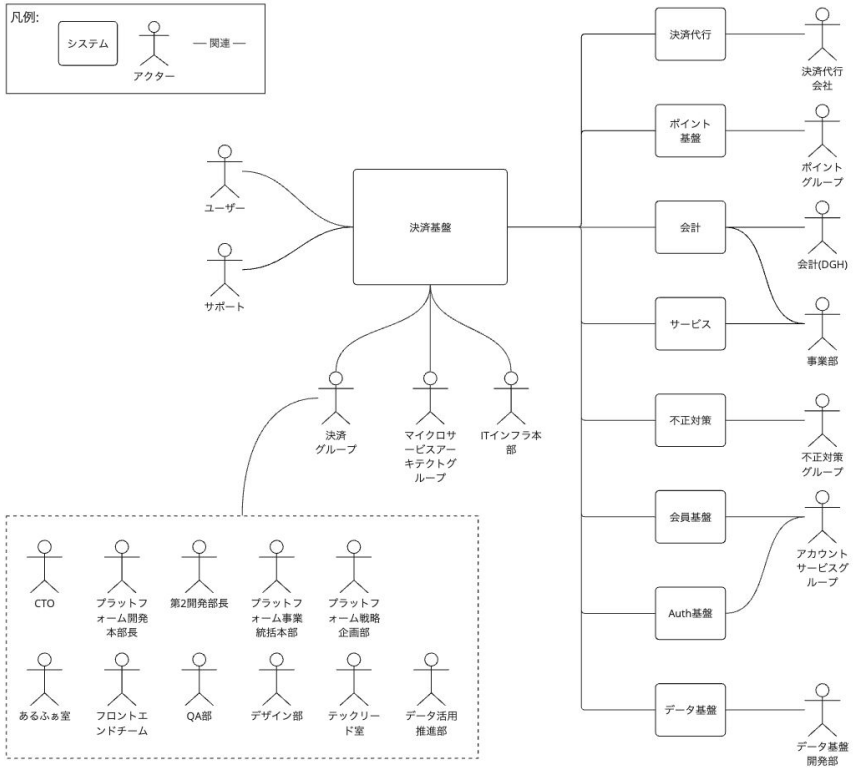


C4モデル

- C4は、コンテキスト、コンテナ、コンポーネント、コード
- システムを、コンテナやコンポーネントに分解してモデリング
- 抽象度を統一して記述する
- 関心事の異なるステークホルダーとの共通認識を築きやすい
- https://c4model.com/

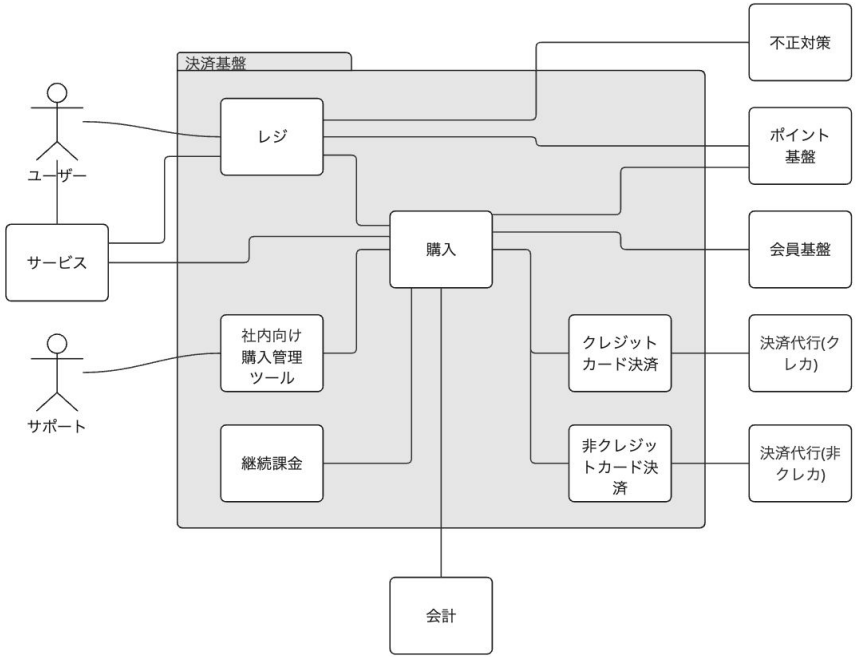
コンテキストモデル

- 決済基盤に関連するアクターとシステムをモデリング
- アクターはステークホルダー
- 技術的負債の返済に直接的、間接的に関連する関係を可視化
- 決済基盤とそれ以外を明確にすることでスコープが分かる



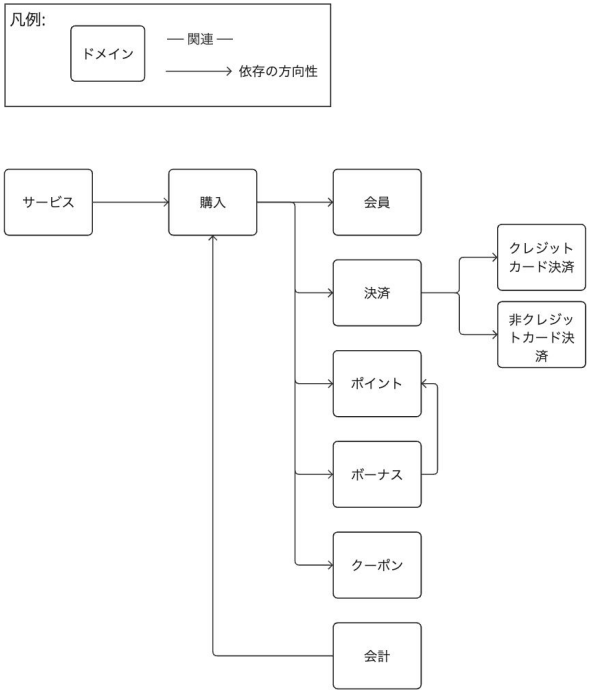
コンテナモデル

- 決済基盤内部のコンテナをモデル化
  - 右図はコンテキスト寄り
  - 実際の図を貼ると細か過ぎる
- EC2やEKS、Auroraなどアプリケーションの実行環境と永続化装置などを書く



コンポーネントモデル

- 物理的な制約を取り除いてコンポーネント間がどのように関連し合っているかのモデル
- *境界付けられたコンテキスト* とか *集約* を意識する
- 実体は相互依存している
- この粒度になると理想が見えて混じりだす



コード

- as-is においては、コードを読むがモデルにしない
- しないというかできない
- 典型的な手続き型で記述されている
- ここの分析に時間をかけない
- コードはこのあと何回も読む
- ここではそこそこにしておいて as-is を言語化する

as-is と to-be を言語化

- as-is を分析しモデリングし、それを元にステークホルダーと対話
- 対話の中から to-be に成る情報を集める
- 発散 と 収束 を繰り返す
- モデルは対話のブレを抑える
- 今の話はこのモデルのここのことですよね

決済基盤の as-is

- 現在の決済基盤は、システム間（決済内外共に）の依存関係が複雑になっている
- 会員、サービス、購入、決済、クーポン、ポイント、ボーナス、会計との関連が相互に依存し合い、責務の境界も曖昧
- 特に、購入ドメインは、関連する他ドメインが本来持つべき責務を担っており、システム的にもチーム間コミュニケーション的にも、必ず間に入る必要がある

決済基盤の to-be

- ドメインモデルに基づいたアーキテクチャを構築
  - 各ドメインの責務が明確で依存関係が整理されている
  - 独立した運用と成長が可能で多方面における速度を向上
- リアクティブなシステムを構築
  - 高い即応性とユーザーに対する効果的なフィードバック
  - 機会損失を削減し、収益への影響を減らす
- SRE文化の醸成と運用コストの削減
  - 無駄な手作業や運用負荷を削減する余裕と習慣を持つ
  - 効率的かつ持続可能な体制を持つ

to-be を実現するとどうなるのか

- 期待値を言語化しておく
- この期待値が指標につながる
- 定性的か定量的かは気にしない
  - 購入に集中しているコミュニケーションが減る
  - コミュニケーションが減るとは
    - サービス開始時のコミュニケーション
    - 障害発生時のコミュニケーション
    - 支払い方法追加時のコミュニケーション
  - 新しい価値を追加する、既にある価値を高める、に時間を使える
- 計測のタイミングを見定める
  - 最初から拘り過ぎず機会を伺い、常に狙う

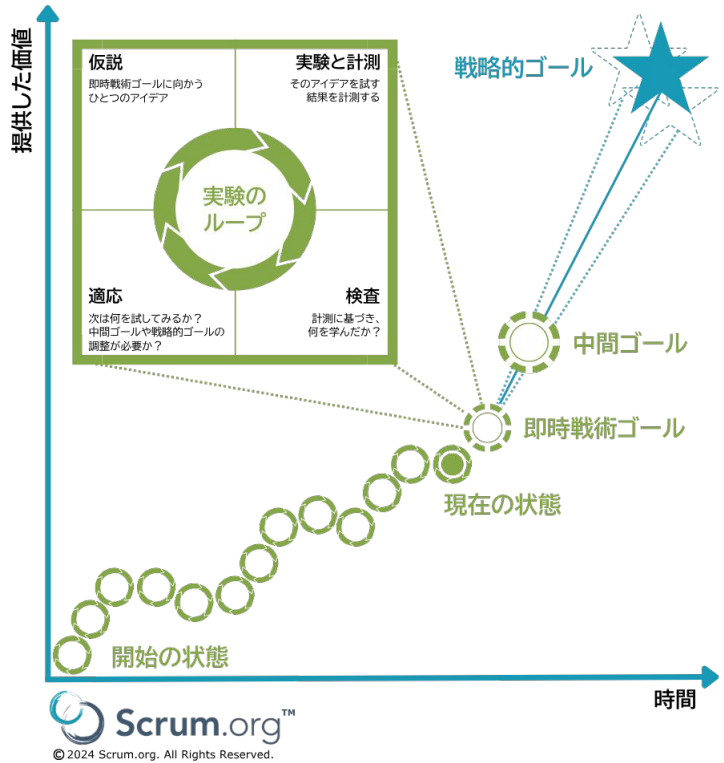
課題と打ち手

- 課題は山程でてくる（粒度も様々）
- すべての課題を一気に解決することはできない
- 段階的に解決していく戦略が必要
- 戦略の立案に エビデンスベースドマネジメント（EBM）を使う

https://www.servantworks.co.jp/resources/translated/evidence-based-management-guide-japanese/

戦略的ゴール

- to-be を実現するために達成したい事柄を出す
- 何をしたら to-be を実現したと言えるかを問う
- 遠いところから手前にブレークダウンするも良し
- 手前から積み上げて遠くを見るも良し



引用元 https://www.servantworks.co.jp/resources/translated/evidence-based-management-guide-japanese/

中間ゴール

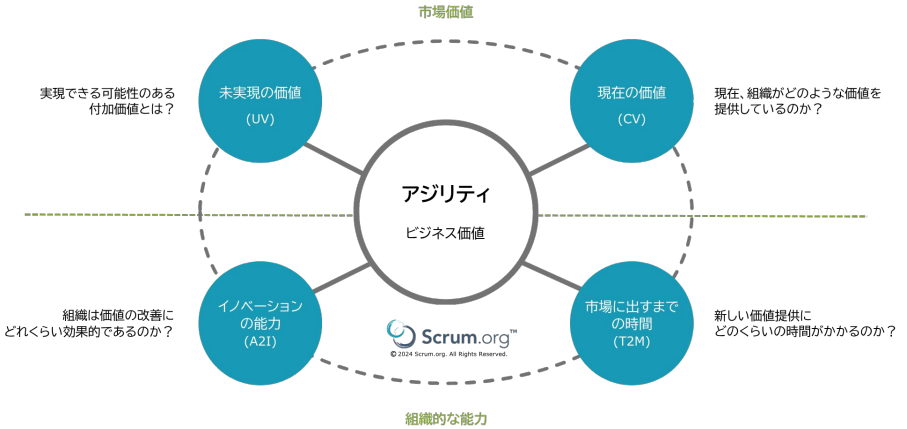
- 今期が終わったときにどういう状態になっていたいかを構想する
- 前後期に分けて考えてもいい
- ゴールを達成したと言えるに必要な指標を出す
- 定性的、定量的、充分、不充分は、いったん置いておく
- 指標は後から足してもいい

即時戦術ゴール

- スクラムで言うならスプリントゴール
- このスプリントで何を達成したいか
- チームで合意して目指していく
- 完了条件を計測
- スプリントレビュー、スプリントレトロスペクティブで検査
- 次のスプリントで適応

得たい価値は何か

- 重要価値領域を参考にする
- 技術的負債の返済は、市場価値を高めない
- 組織的な能力の向上が得たい
- このことを確認しながら各ゴールに向かえているか経験的に進んでいく



引用元 https://www.servantworks.co.jp/resources/translated/evidence-based-management-guide-japanese/

戦略的ゴール

- システムのモダナイゼーション
  - 依存関係の整理
    - 最も複雑な購入ドメインから構造を再構成
    - 購入ドメインからの影響を他のドメインに波及し全体を最適化
  - 障害に対する耐性の向上
    - 障害を前提とした設計
    - 状態の整合性を自動で回復する仕組みを構築
- 運用体制の強化
  - サービスレベル目標 (SLO) に準拠した運用
    - エラーバジェットアラートに基づいた障害対応
    - オブザーバビリティを獲得しトレース可能な状態を構築
  - Four Keys に基づく改善サイクル

中間ゴール - 2025年度

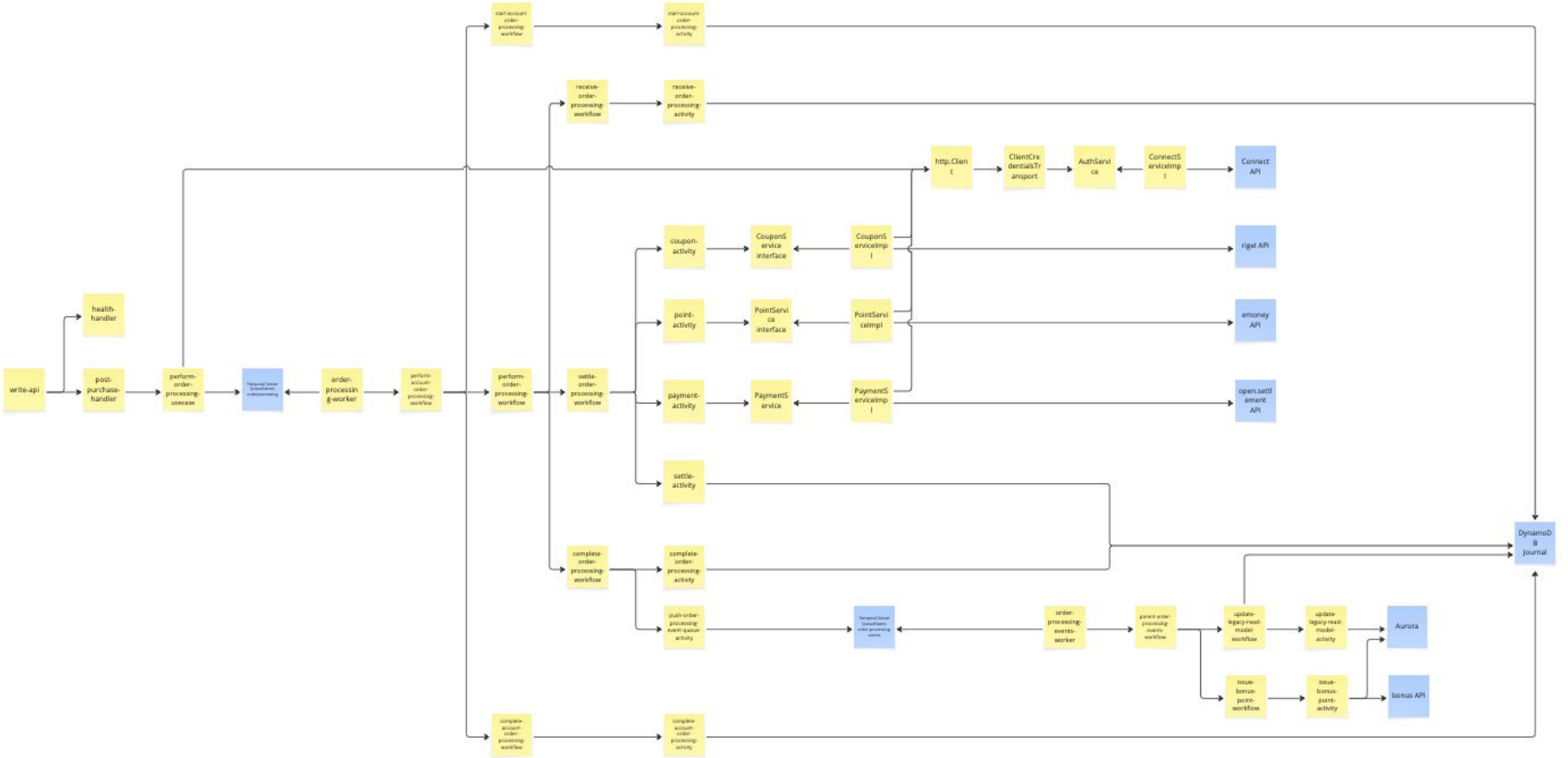
- 上期
  - 購入システムの一部をモダン化し、テスト運用
    - 最小のユースケースを対象
- 下期
  - モダン化した購入システムをテスト運用から本格運用へ
    - ユースケースの拡充
    - 対象ユーザーの拡大
    - 組織の拡大

既存システムの一部をモダン化

- ストラングラーフィグパターンを参考に置き換える
- なぜこのパターンを参考とするか
- Architecture Decision Record (ADR) に書いて合意形成
- コンポーネント図で切り替えステップを可視化
  - ステップ1: Facadeレイヤーの構築
  - ステップ2: モダンシステムを鍛える
  - ステップ3: モダンシステムとレガシーシステムの辻褄を合わせる (CQRS)
  - ステップ4: カナリアリリースで段階的に移行
- 現在、ステップ3を絶賛開発中
  - Claude Code等のAIエージェントがアウトプットを爆速加速

Temporal と CQRS&ES

- 購入ドメインには、受注処理がある
- 受注処理は、各処理工程の成功と失敗がある
- それらの実行状況の管理、工程の組換え、途中からの再開がある
- 状態遷移を定義し、予期せぬ割り込みを防ぎ、着実に実行する
- Actorモデルが適切だがオーバーエンジニアリング
- Temporal と CQRS & Event Sourcing を組み合わせる
- https://temporal.io/


write-api
health-handler
post-purchase-handler
perform-order-processing-usecase
order-processing-worker
perform-account-order-processing-workflow
start-account-order-processing-workflow
start-account-order-processing-activity
receive-order-processing-workflow
receive-order-processing-activity
perform-order-processing-workflow
settle-order-processing-workflow
coupon-activity
CouponService interface
CouponServiceImpl
point-activity
PointService interface
PointServiceImpl
payment-activity
PaymentService
PaymentServiceImpl
settle-activity
complete-order-processing-workflow
complete-order-processing-activity
push-order-processing-event-queue-activity
order-processing-events-worker
parent-order-processing-events-workflow
update-legacy-read-model-workflow
update-legacy-read-model-activity
issue-bonus-point-workflow
issue-bonus-point-activity
complete-account-order-processing-workflow
complete-account-order-processing-activity
http.Client
ClientCredentialsTransport
AuthService
ConnectServiceImpl
Connect API
Figel API
emoney API
open.settlement API
DynamoDB Journal
Aurora
bonus API

ここまで ひとり

- 後付けで濃い内容にしているものの大枠についてはひとりで推進
  - もちろん、たくさんの方々にご協力をいただいている
  - 壁打ち、情報収集、意見交換、フィードバックなどなど
- 入社から3ヶ月で立案
- 2025年01月からチームを立ち上げるべく人員の募集
- 既存チームの横に小さなチームを立ち上げる

2-4名のチーム

- two pizza team より小さいチーム
- トライ & エラー のサイクルを早めたい
- 意思伝達のパスを限りなく少なくしたい
- キレイなチームにしない
- 時にはマイクロマネジメントを辞さない
- 必要ならいつでもオーナーシップ、リーダーシップを奪ってもらう

叩き台を作る

- デファクトとなるモジュールを小さなチームで立ち上げる
- 構造的に堅牢な作りを整える
  - レイヤ化アーキテクチャ
  - 契約プログラミング
  - SOLID原則
  - Tell don’t ask
  - EventSourcing
  - ドメイン駆動設計
  - 腐敗防止層

みんなで育てる

- チームトポロジーとSECIモデルを組み合わせる
  - Enabling Team で共通体験を得て、共同化
  - Stream Aligned Team に入り体験を、表出化
  - チーム内の体験と組み合わせて、連結化
  - 体験を横断的展開ができるよう、内面化
- ダイナミックリチーミングにトライする
  - 我々がどういった自由を求めるか言語化する
  - 生成AIの発展により、バディをAIに求めて、より柔軟なチーム構造を模索

まとめ

- 表面的な混沌さに惑わされず、構造に着目しよう
- C4モデルとEBMは、共通認識を得るのに使う
- ストラングラーフィグパターン と CQRS で段階的にモダン化
- ひとり→2-4名の小チーム→チームトポロジーとSECIモデル
  段階的にモダン化
- 障害耐性↑・リードタイム↓・組織学習↗ で“創る時間”を取り戻す

「小さく切って、測って学び、大きく広げる」

ご静聴ありがとうございました

誰もが 見たくなる 未来。

DMM.com