アーキテクチャレベルで考える開発生産性

2024/06/29(土)
開発生産性Conference 2024

合同会社DMM.com
ミノ駆動

プラットフォーム開発本部 第3開発部 DeveloperProductivityGroup

自己紹介

ミノ駆動（ @MinoDriven ）

合同会社DMM.com

プラットフォーム開発本部 第3開発部

DeveloperProductivityGroup

DMMプラットフォームの設計を改善し開発生産性向上を図るのがミッション

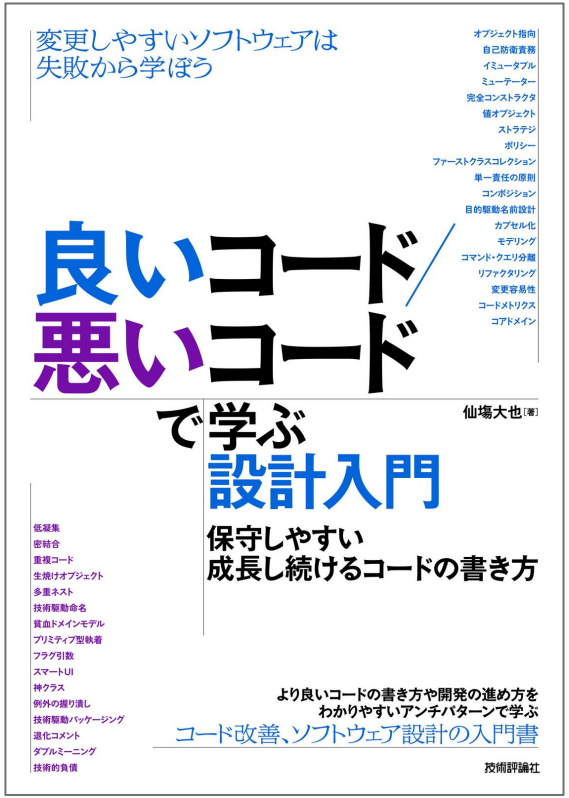
著書紹介

『良いコード／悪いコードで学ぶ設計入門』

変更容易性の高い設計を学ぶ、初級～中級向け入門書

ITエンジニア本大賞2023技術書部門大賞受賞

11刷重版



本セッションの理解目標

- ソフトウェアの開発生産性を高めるには、開発の自動化やプロセス改善だけでなく、アーキテクチャの設計も重要であること。
- サービスの競争優位性が何であるかを把握し、競争優位性を高められるアーキテクチャを設計すること。
- こうしたアーキテクチャの設計には、ドメイン駆動設計の戦略的設計が有用であること。

突然ですが…

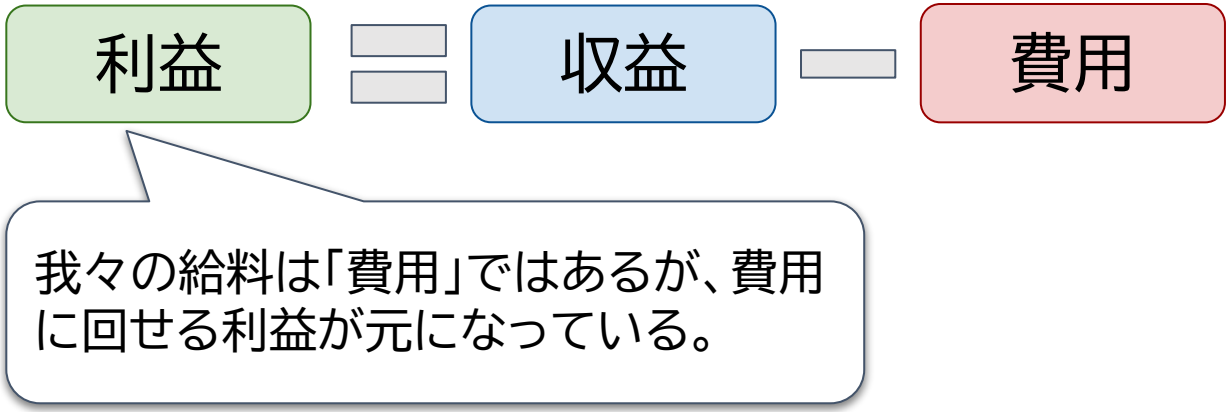
皆さんのお給料、
どこから出ていますか？

利益、収益、費用の関係

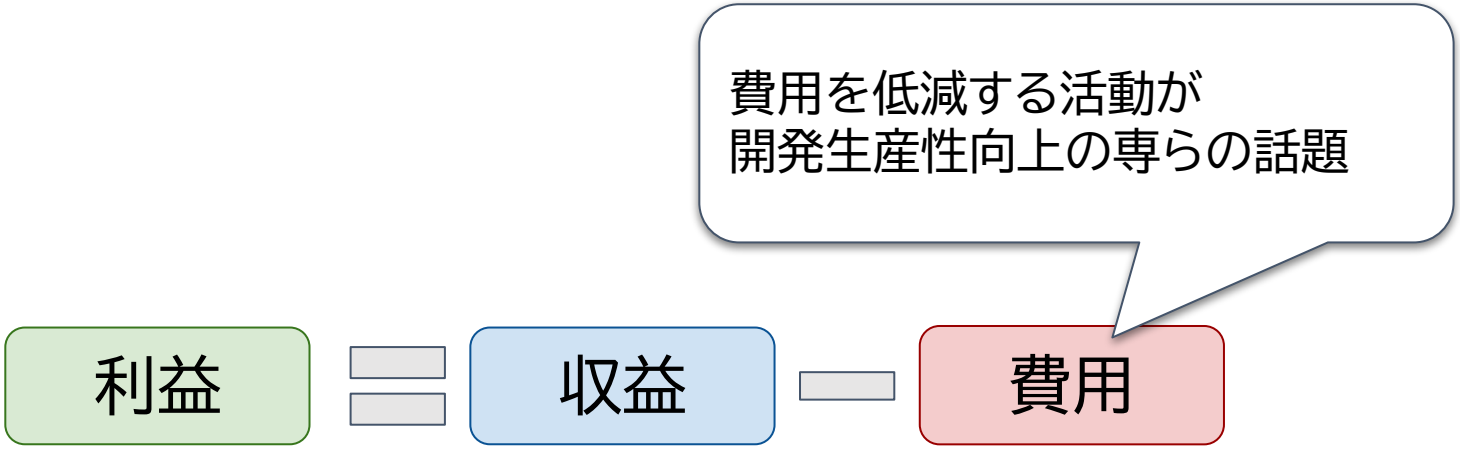
費用：業務遂行にかかる諸々のコスト

収益：商品やサービスが稼いだお金の総額。費用が差し引かれる前の金額。

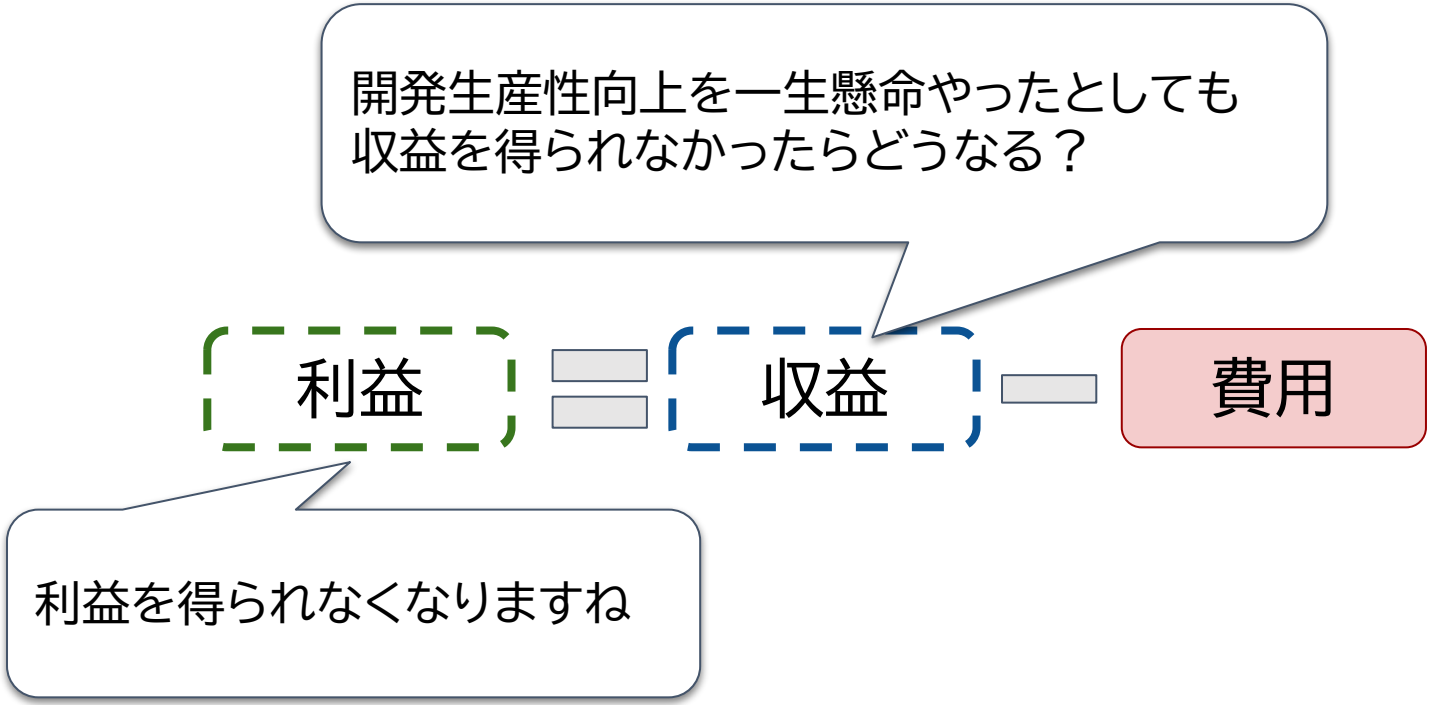
利益：収益から費用を差し引いた金額。



開発生産性の関わり



収益が得られなかったらどうなる？



極端な話

- **CI/CD環境、Four Keys計測、チャットツールやタスク管理ツールなど開発生産性向上の環境整備は完璧！**
- **でも顧客が見向きもしない、役に立たないサービスが完成しました……**

いくら頑張って開発生産性向上の環境を整備しても、収益を出せなかったら開発生産性も何もあったものではないですね
（※開発生産性向上活動は重要です、否定はしません）

だから収益を上げられるサービスを作りましょう
「収益向上」をベースに開発生産性を考えましょう
収益と開発生産性を高めやすいアーキテクチャを考えましょう

本日はこういうお話をします

ところで

あなたが開発するサービス、
業界で独壇場ですか？

そうじゃないですよね
ほとんどの場合
競合他社がいますよね

競合他社がひしめき合っている状況で
生半可な機能開発で他社に勝てますか？
勝てませんよね

じゃあ何を開発しますか？

顧客があまり触らないような
サブ機能やオプション機能を
集中的に改善しますか？

それともすでにコモディティ化している
機能を改修しますか？

違いますよね

他社が追従できない、差別化可能な、
「ここがウリだ！」と言えるような
競争優位になる魅力的な機能を
開発しますよね

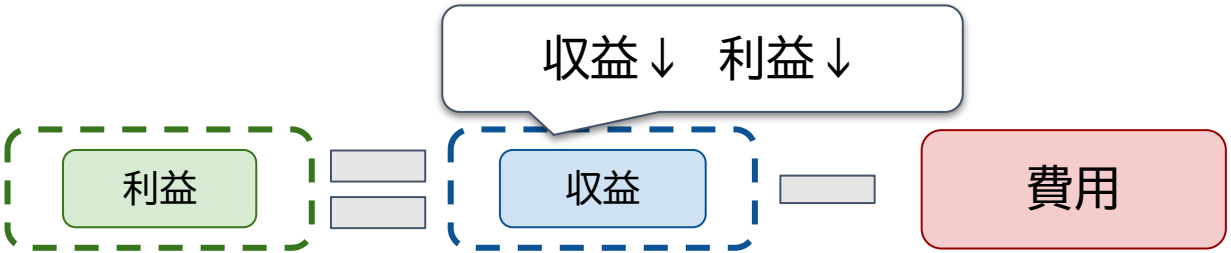
ではあなたが開発するサービスの
競争優位性が何か
すぐに説明できますか？

競争優位性が何かよく分かっていなくて
機能開発が五月雨になってたり
していませんか？

何の話をしているかというと…

機能性

- ソフトウェア品質特性のひとつ。機能が顧客ニーズを満たす度合い。
- 機能性が高いと当然収益も上がる。
- 競争優位性のある機能の開発が収益向上する上で重要。
- 競争優位性を意識しないと思いつきで五月雨な開発になりがち。
- 逆噴射して「なんでこんな仕様に変えたの？」と顧客を失望させることも。
- 優位性のない機能は顧客を満足させることができず、収益が低下。



もうひとつ別の観点の話をします

では何が競争優位な機能か
わかっているものとします

もしくは頻繁に仕様変更される機能を
思い浮かべてください

その機能のためのロジックが
ソースコードのどこにあるのか熟知していますか？

どこにあるのか把握が困難で
いちいち詳しく読みにいかなければ
ならなくなっていませんか？

その機能のソースコードは
変更しやすい構造になっていますか？

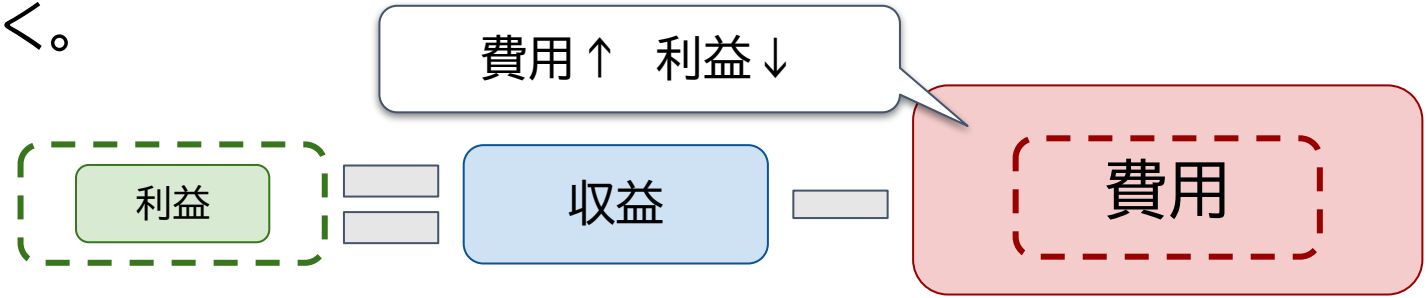
混乱して複雑化していて
変更が難しい構造になっていませんか？

そのコードを変更すると
別の機能が壊れたりしませんか？

何の話をしているかというと…

変更容易性

- ソフトウェア品質特性のひとつ。
- なるべくバグを埋め込まず素早く正確に変更可能な度合い。
- 変更容易性の高い構造であるほど開発コスト(費用)が下がる。
- 逆に複雑で混乱した構造は変更容易性が低く、開発コスト増大。
- 特に**競争優位になる魅力的な機能は顧客ニーズが多く、頻繁に仕様変更され、放っておくとどんどん複雑化していく。**
- 機能の魅力をもっと高めようにもなかなか変更できなくなり、優位性が失われていく。



費用に関してもうひとつ重要なこと
「費用は有限」

選択と集中

- 当たり前ですが開発リソースは有限です。無限ではありません。
- だからサービスの全部に対して等しく開発投資するなんてことは不可能です。
- 優先度を決め、重要な部分に集中的に開発リソースを投じなければならない現実があります。
- ではどの部分に集中するか？それは**競争優位性を発揮できるビジネス領域であり機能**ですね。
- 変更容易性の設計も同様に、重要な部分に集中すべきです。

重要部分の機能性と変更容易性を高めることで
利益が上がる
投資費用対効果すなわち
開発生産性が上がる

ここまで一旦まとめ

1. 限られた開発リソースの中で、
2. 競争優位性を発揮できるビジネス領域を見定め、
3. その領域に対して集中的に開発リソースを投じ、魅力的な機能を開発する
4. 魅力的機能の機能性を素早く高められるように、その機能に対して変更容易性の設計コストを集中させる

では
機能性と変更容易性を高めるために
どう取り組めば良いのでしょう？

ドメイン駆動設計

ドメイン駆動設計には
機能性と変更容易性を高め
持続的に利益向上していくための戦略や
アーキテクチャ設計のノウハウが
網羅されています

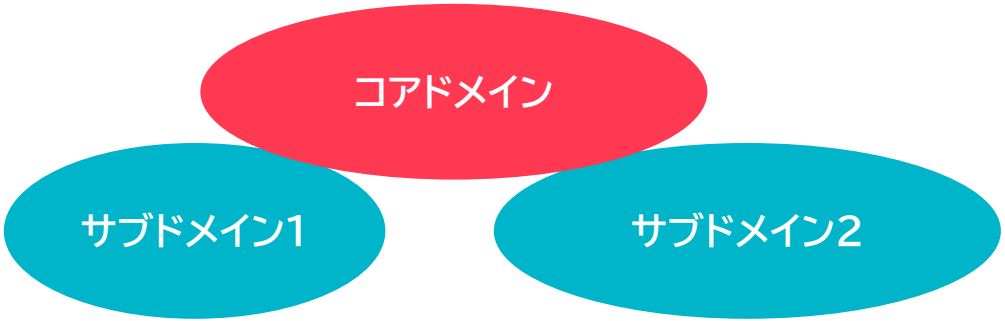
ドメイン駆動設計における
「戦略的設計」
のノウハウを
いくつか紹介します

ドメイン駆動設計の「戦略的設計」は
きわめて重要ではあるものの
あまり話題にされないので
概要だけでもしっかりおさえておいてほしいです

コアドメイン

機能性

- ドメインとは、ソフトウェアが解決する事業領域。
- コアドメインとは、**差別化が図られ、競争優位性を発揮する領域。**
- 一方で必要ではあるがコアではない領域をサブドメインと呼ぶ。
- ドメインエキスパート（事業課題の専門家）と話し合い、コアとサブを見分けることが重要。**これが一番大変！**
- ちなみにソフトウェアはユーザー目的を満たすように作られるものなので、コアを**主目的**、サブを**副目的**と個人的に呼称していたりする。



ドメインビジョン声明文

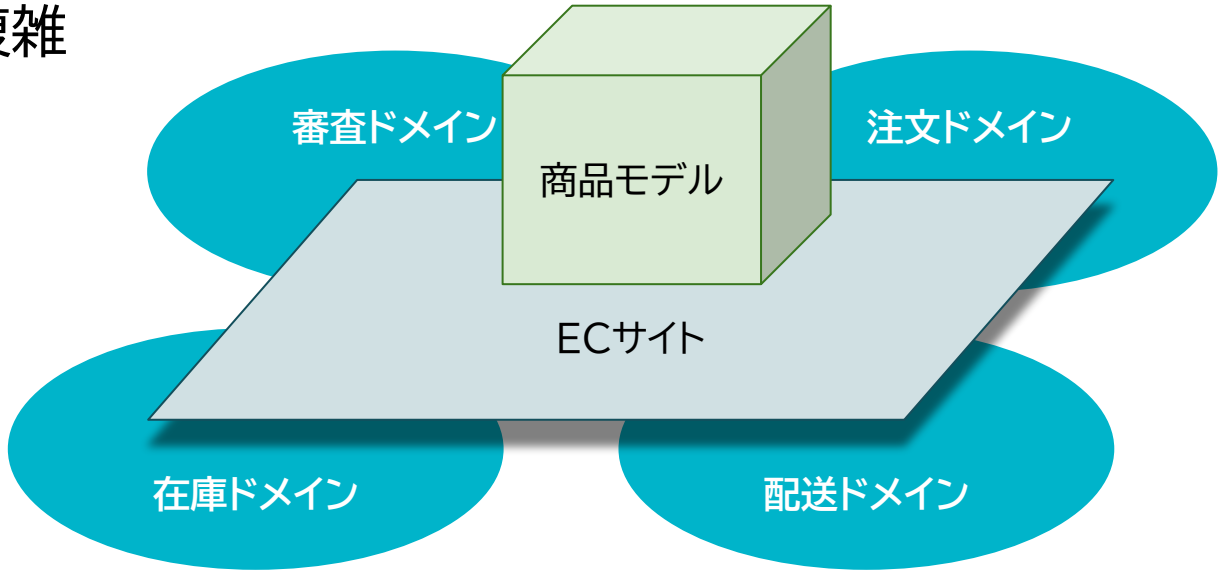
機能性

- コアドメインを見定めたらそのままにしておくのではなく、関係者全員が共通認識を持てるように指針を示す必要がある。
- ドメインビジョン声明文とは、コアドメインがもたらす価値を簡潔に記述したドキュメント。

境界付けられたコンテキスト

変更容易性

どのドメインでも共通の、統一的な一枚岩モデルにすると モデルが多目的に使われ、巨大化複雑化する。



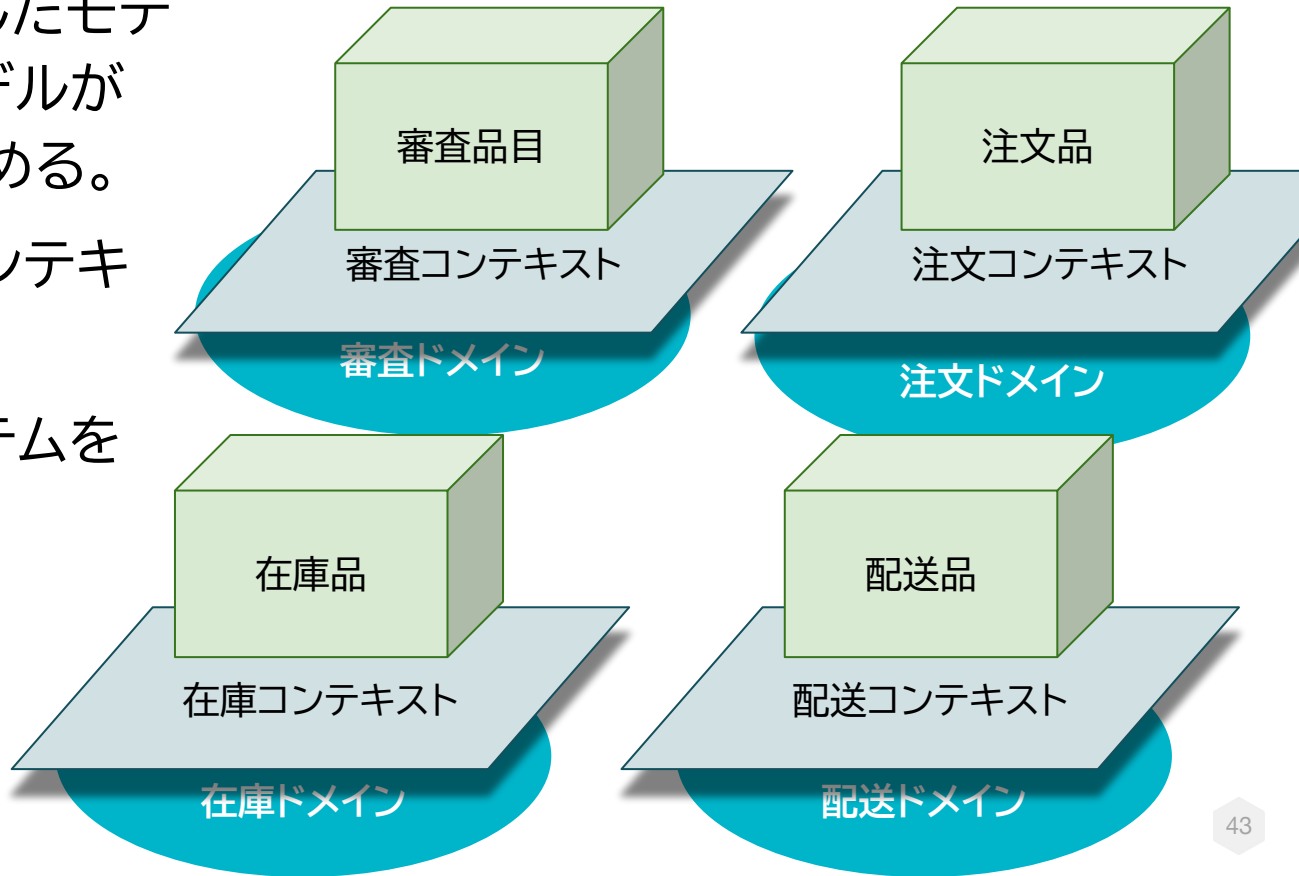
境界付けられたコンテキスト

変更容易性

各ドメイン(目的)に特化したモデルを設計し、各特化型モデルが適用可能なスコープを決める。

これが境界付けられたコンテキスト。

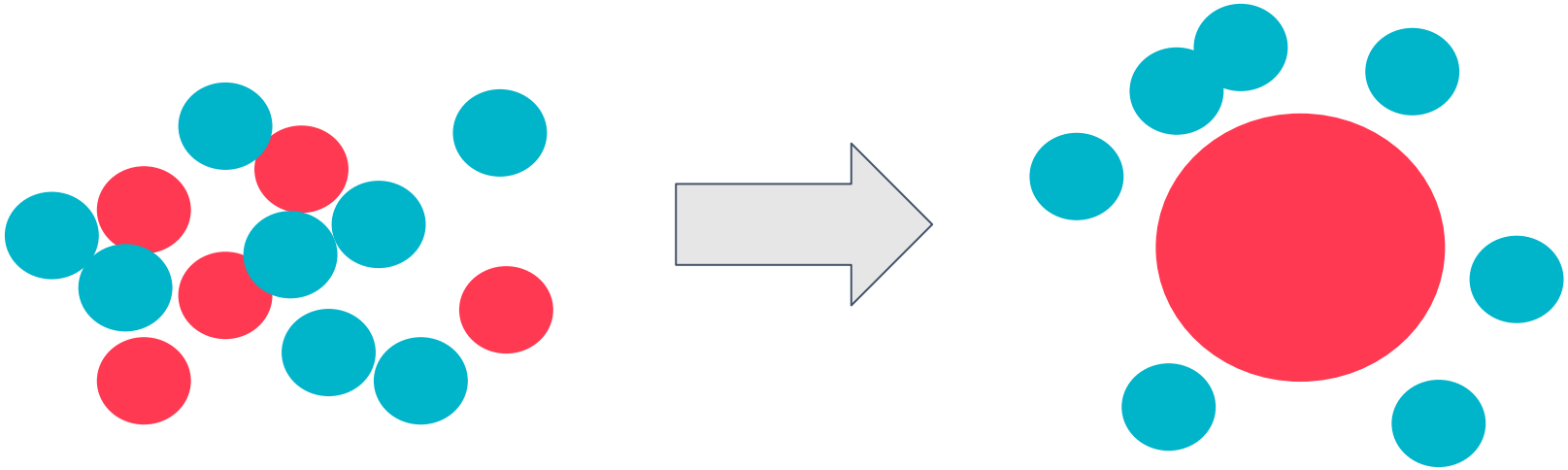
後述のコアとサブでシステムを分けるのにも用いる。



蒸留

混ざりあったコンポーネントを分離し、価値のある部分を抽出するプロセス。DDDではもっぱらコアドメインの要素を抽出することを意味する。

ばらばらに断片化したコアロジックを抽出し、コア用のモジュール(サブシステム)に集めてカプセル化する。



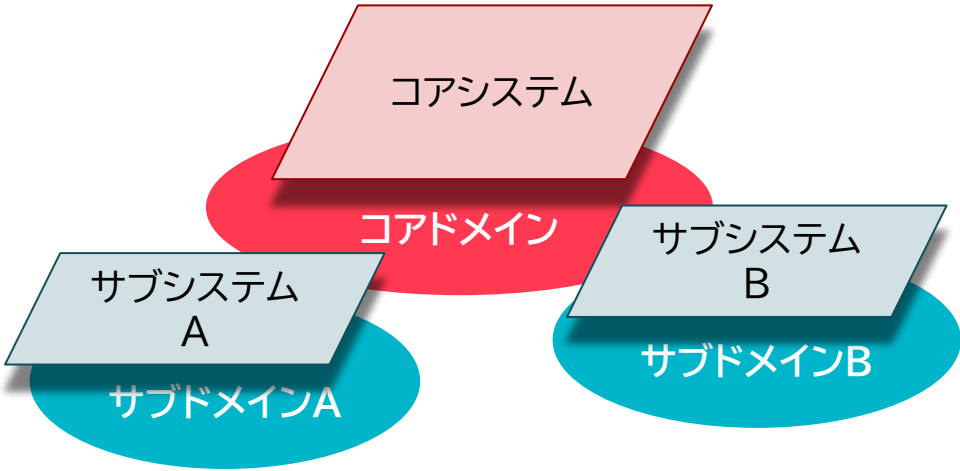
隔離されたコア

変更容易性

コア用のロジックとサブ用のロジックが複雑に絡み合っていると、コアの機能を改善しようにも変更が難しくなってしまう。

従って、境界付けられたコンテキストと蒸留に基づきコアとサブでシステムを分離隔離する。

コアにサブの要素が入り込まないよう徹底的に純化し、コアに特化したシステムにする。

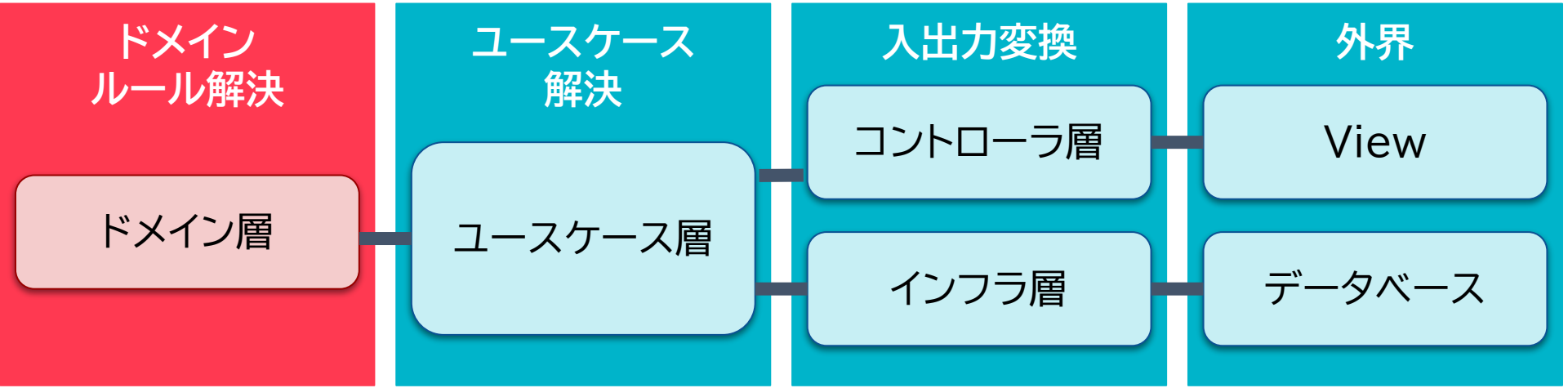


レイヤードアーキテクチャ （※戦術的設計）

変更容易性

下図はレイヤードアーキテクチャの一例(いろいろバリエーションあり)。

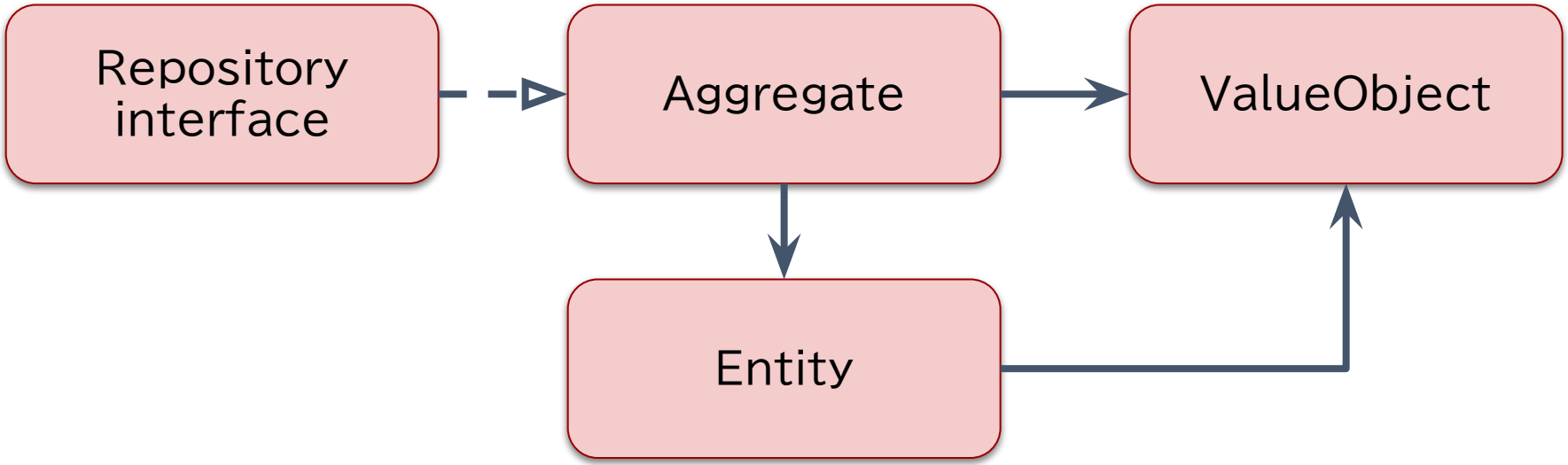
ドメイン関心事や技術関心事単位で関心を分離することで変更容易性の向上を図る。



ドメイン層（※戦術的設計）

変更容易性

以下、代表的な設計パターン(このパターンには限らない)を用いてドメイン概念を整理することにより変更容易性が向上する。



コアへの開発投資を優先する

- 開発リソースは有限、全てに対して満遍なく等しく開発投資することは不可能。
- エンジニアによってスキルの熟達度はそれぞれ。
- 同様に変更容易性の設計スキルもピンキリ。
- サービスの競争優位性を高めるには、コアに対して優先的に開発コストを割き、スキルの高いエンジニアをアサインする必要がある。
- サブドメインの機能については、開発を外部に委託する、サードパーティのサービスやツールで賄う、といった戦略も考えられる。

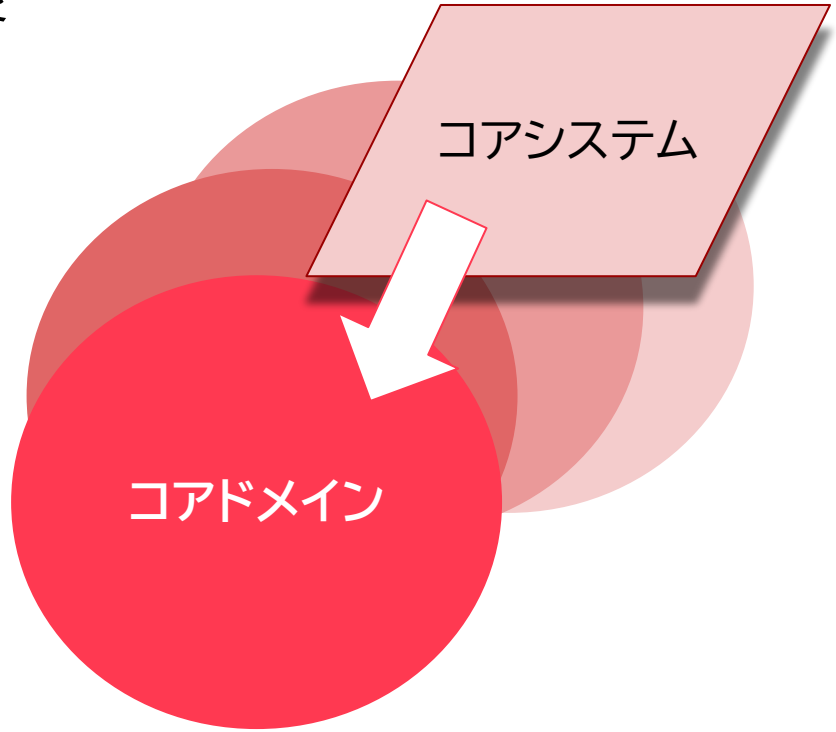
コアドメインは移動する

変更容易性

ユーザー要求の変化、競合他社の台頭、優位性のコモディティ化など外的な要因により、競争舞台であるコアドメインは時代とともに移動する。

ここでコアシステムの構造が複雑で粗悪だと、コアドメインの変化に追従できなくなる。

追従できるようコアシステムの変更容易性を高めておくことが、競争優位性を維持する上で肝要。



我々はアジャイルを実践できているのか

「アジャイル（素早い、機敏）」という言葉が放たれるとき、開発プロセスや組織、FourKeysといった、システム本体ではなくシステムを取り巻く世界の話がよく挙がります。

しかし、「価値」が構造化されていないシステムは、アジャイルと言えるでしょうか？

複雑怪奇で変更が困難なシステムは、アジャイルと言えるでしょうか？

価値を素早く成長し続けられる構造を設計することが、アジャイルな開発を推進し、開発生産性を向上する上で重要な位置付けであると考えます。

ご清聴ありがとうございました