#アーキテクチャcon_findy

ソフトウェア設計の課題・原則・実践技法

2025年11月21日

有限会社システム設計　増田 亨

自己紹介

増田　亨（masuda220）

専門領域

- 業務系アプリケーションの開発

最近の仕事

- 大きな泥団子退治のお手伝い
- エンジニアの設計スキル向上のお手伝い

著書(2017)

＊1

訳書(2024)

＊2

*1 増田 亨(2017) 『現場で役立つシステム設計の原則』技術評論社
*2 Vlad Khononov(著) 増田 亨、綿引 琢磨(訳) 2024 『ドメイン駆動設計をはじめよう』オライリージャパン

お話する内容

① ソフトウェア設計の課題

② 良い設計を生み出すための基本原則

- ✓ 変更容易性に焦点を合わせる
- ✓ 事業活動を理解して設計判断する

③ ソフトウェア設計の実践技法

- ✓ 乱雑なコードの整理整頓

（参考）戦略的なデータマネージメントの実践技法

①ソフトウェア設計の課題

ソフトウェア開発の目的

ソフトウェアシステムは
事業活動を効率的かつ効果的に進めるための
さまざまな仕組みの一つ

ソフトウェアシステムを開発し運用する目的は
高業績を持続させること

事業活動とソフトウェアシステムの連動性

広く連動する

あらゆる事業活動のデジタル化が進んでいる

深く連動する

- 旧来：データのCRUDと単純な加工（状況判断と行動は人間）
- 現在：状況判断・推論・決定をソフトウェアシステムに組み込む

双方向に影響する

- 事業活動の変化がソフトウェアシステムを変化させる
- ソフトウェアシステムの変化が事業活動を変化させる
- 影響する範囲が広く深い → **相互作用しながらともに発展していく**

ソフトウェア設計の課題

事業活動と広く深く連動するソフトウェアは**複雑**である

事業活動の変化と連動しソフトウェアは**変化を繰り返す**

事業活動と同様に**未知の課題**に**不確実な状況**で取り組む

あらゆる事業活動と同様に、

設計に使える時間と資源は限られている

②良い設計を生み出す基本原則

良い設計を生み出す２つの原則

変更容易性に焦点を合わせる

事業活動を理解して設計判断する

変更容易性に焦点を合わせる

設計とは何か？

ある目的のために、

形の無いところに形を与え、

その形を変え続ける活動

良い設計は悪い設計よりも変更しやすい

『達人プログラマー ―熟達に向けたあなたの旅』第２版

Andrew Hunt(著)David Thomas(著)村上雅章(訳)　2020年　発行：オーム社

セクション８　よい設計の本質　Tips14

より実践的なソフトウェア設計の本質

整理整頓されたコードは

乱雑なコードよりも

変更が楽で安全である

良い設計と悪い設計

整理整頓されたコード

変更が楽で安全

乱雑なコード

変更が厄介で危険

事業価値を生み出す

事業に損失をもたらす

事業活動を理解して設計判断する

事業活動を理解して設計判断する

設計改善（整理整頓）に**使える時間**は限られている

ソフトウェアシステム全体の
どこを、いつ、どこまで整理整頓するか

費用対効果の高い個所を**事業視点で判断**する

事業戦略とソフトウェアの実装を結びつける

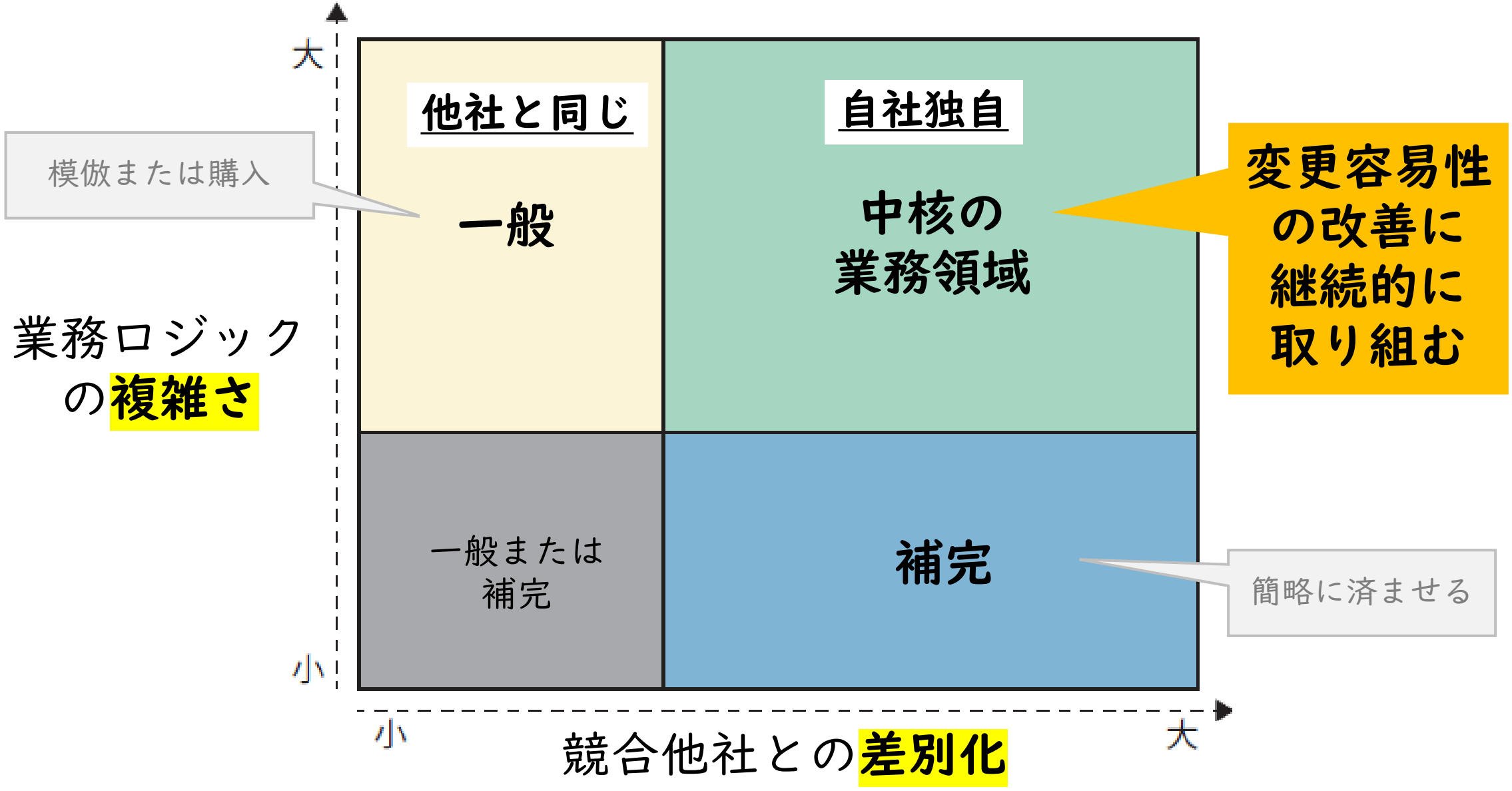
- ✓ 高業績を持続させるための**事業戦略（差別化戦略）**と**ソフトウェアの実装**を結びつける

- ✓ **差別化戦略に適合**する、ソフトウェアの**設計方針**と開発の**優先順位**を見つける

- ✓ 差別化への影響が少ない個所は簡略に済ませる（やらない選択）

事業活動を理解して設計するための参考図書

【エッセンシャル版】


大
他社と同じ
自社独自
模倣または購入
一般
中核の
業務領域
変更容易性
の改善に
継続的に
取り組む
業務ロジック
の複雑さ
一般または
補完
補完
簡略に済ませる
小
小
大
競合他社との差別化

業務領域の分類と開発方針の違い

	中核	一般	補完
競争優位性	◎	×	○
複雑さ	◎	○	×
変化	◎	×	△
開発方針	**独自開発**	模倣または 購入	CRUD/ETLの 簡易開発

③ソフトウェ設計の実践技法

再掲

良い設計を生み出す２つの原則

変更容易性に焦点を合わせる

事業活動を理解して設計判断する

良い設計を生み出す２つの原則

変更容易性に焦点を合わせる

事業活動を理解して設計判断する

どう実践するか？

設計改善の費用対効果を最大にする

小さな設計改善を開発組織全体で**継続する**

戦略的に重要な個所を**重点的に**設計改善する

設計改善の費用対効果を最大にする

小さな設計改善を開発組織全体で**継続する**

戦略的に重要な個所を**重点的に**設計改善する

設計改善に使える時間は限られている

実践的で効果的なアプローチ

- ✓ 機能修正や機能追加の**重要度を事業視点で判断**する
- ✓ 重要な機能修正や機能追加の**対象範囲に限定**して
  乱雑なコードを**整理整頓**する
- ✓ 乱雑なコードを整理整頓することで、
  変更が楽で安全になった**効果を検証**する

ソフトウェア設計のアンチパターン

- **初期の設計**（情報不足の設計）に時間をかける
- 事業的に**重要度が低い場所の設計改善**に時間をかける
- 乱雑なコードを**整理整頓しないで**機能を修正追加する
- **機能の修正追加が不要な個所**を整理整頓する

設計改善（コードの整理整頓）三つのレベル

- **小さな設計改善**
- **大きな設計改善**
- **戦略的な設計改善**

- 三つのレベルを並行して進める（段階論ではない）
- 下に行くほど、効果がはっきりするのに時間がかかる
- 下に行くほど、良い設計か悪い設計か判断が難しい

小さな設計改善　初級レベル

① 不要なコードを削除する（ノイズを減らす）

- コメントアウトされたコード
- 使われていない（であろう）コード
- 余計なコメント、間違っている（であろう）コメント

② チャンキング（異なる関心事の境界を見つける）

- 改行を使って、式や文を複数行に分ける
- 改行を使って、多数のフィールド変数や多数の文をグループ分けする

③ グループ分けした単位に名前を付ける

- 説明用変数を使う（ 例：boolean isValid = 数量 > 0 ）
- メソッドに抽出する（ 例 boolean isValid() {return 数量 > 0;}

小さな設計改善　中級レベル

関連するロジックとデータを一つのクラスに集め（カプセル化）、
クラス名とメソッド名で意図を説明する

① 算術演算/比較演算/論理演算のカプセル化　→値オブジェクト

プリミティブなデータ型とそれを使った演算を一つのクラスに集める

② コレクション操作のカプセル化　→コレクションオブジェクト

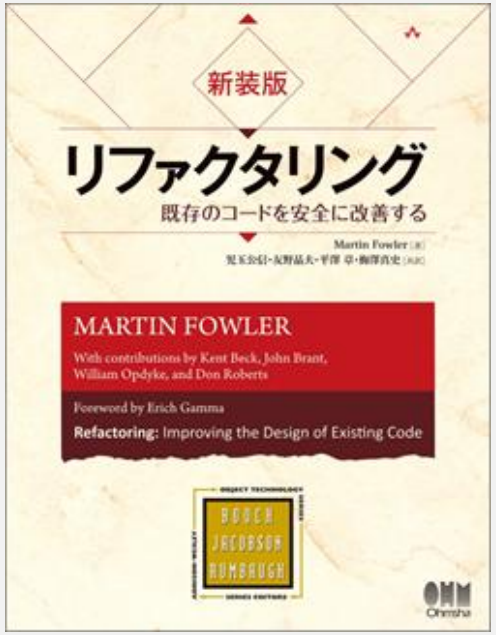
コレクション型のデータとそのループ処理を一つのクラスに集める

③ 区分を使った条件分岐のカプセル化　→区分オブジェクト

条件ごとの定数やロジックをenumクラスに集めて整理する

小さな設計改善の参考図書

第Ⅰ部　整頓
第Ⅱ部　管理術



第3章 コードの不吉な臭い
４章～１０章

１章 ちいさくまとめてわかりやすく
２章 場合分けのロジックを整理する

大きな設計改善（小さな改善を積み重ねる方向）

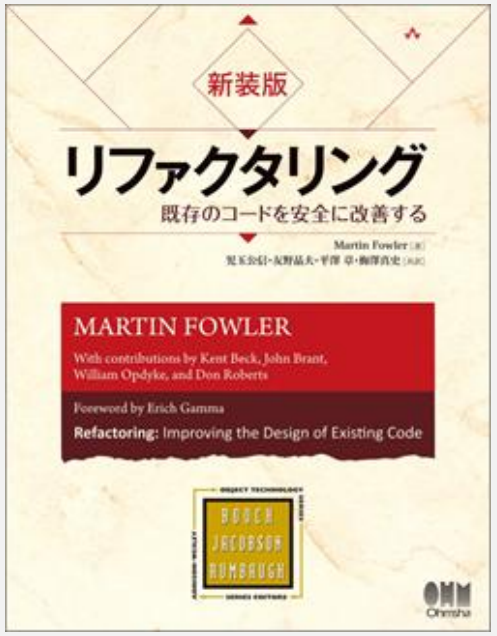
① 計算判断・出力・入力の三つの関心事をクラスとパッケージを使って切り離す

a. 出力のデータ構造に影響された計算判断クラスを作らない
b. 入力のデータ構造に影響された計算判断クラスを作らない
c. 出力クラス、入力クラスに計算判断ロジックを持ち込まない

② アプリケーション特化のデータ型（値オブジェクト、コレクションオブジェクト、区分オブジェクト）を使って計算判断ロジックを記述する

計算判断ロジックを記述するクラスでは、プリミティブなデータ型(int, String, LocalDate, …)とプリミティブな操作を隠蔽する

大きな設計改善の参考図書



第12章 大きなリファクタリング

3章 業務ロジックをわかりやすく整理
4章 ドメインモデルの考え方で設計する

第III部 深い洞察に向かうリファクタリング

戦略的な設計改善（戦略的なコード整理）

- ✓ **事業戦略**と**ソフトウェアの実装**を結びつける

- ✓ **差別化戦略に適合**する、ソフトウェアの**設計方針**と開発の**優先順位**を見つける

- ✓ 差別化への影響が少ない個所は簡略に済ませる（やらない選択）

事業戦略とソフトウェアの実装を結びつける

差別化戦略

競合他社と異なる
自社独自の価値提案

事業戦略とソフトウェアの実装を結びつける

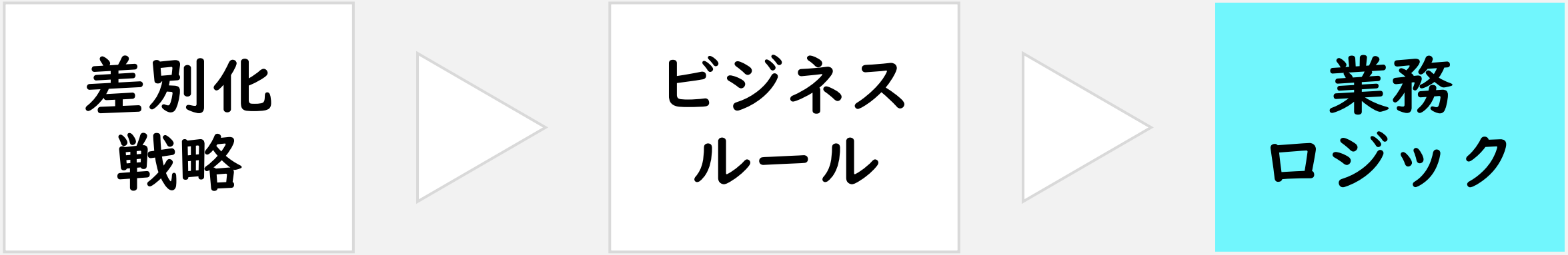
差別化戦略

ビジネスルール

競合他社と異なる
自社独自の価値提案

差別化戦略の実行手段
適切な行動を刺激
不適切な行動を制限

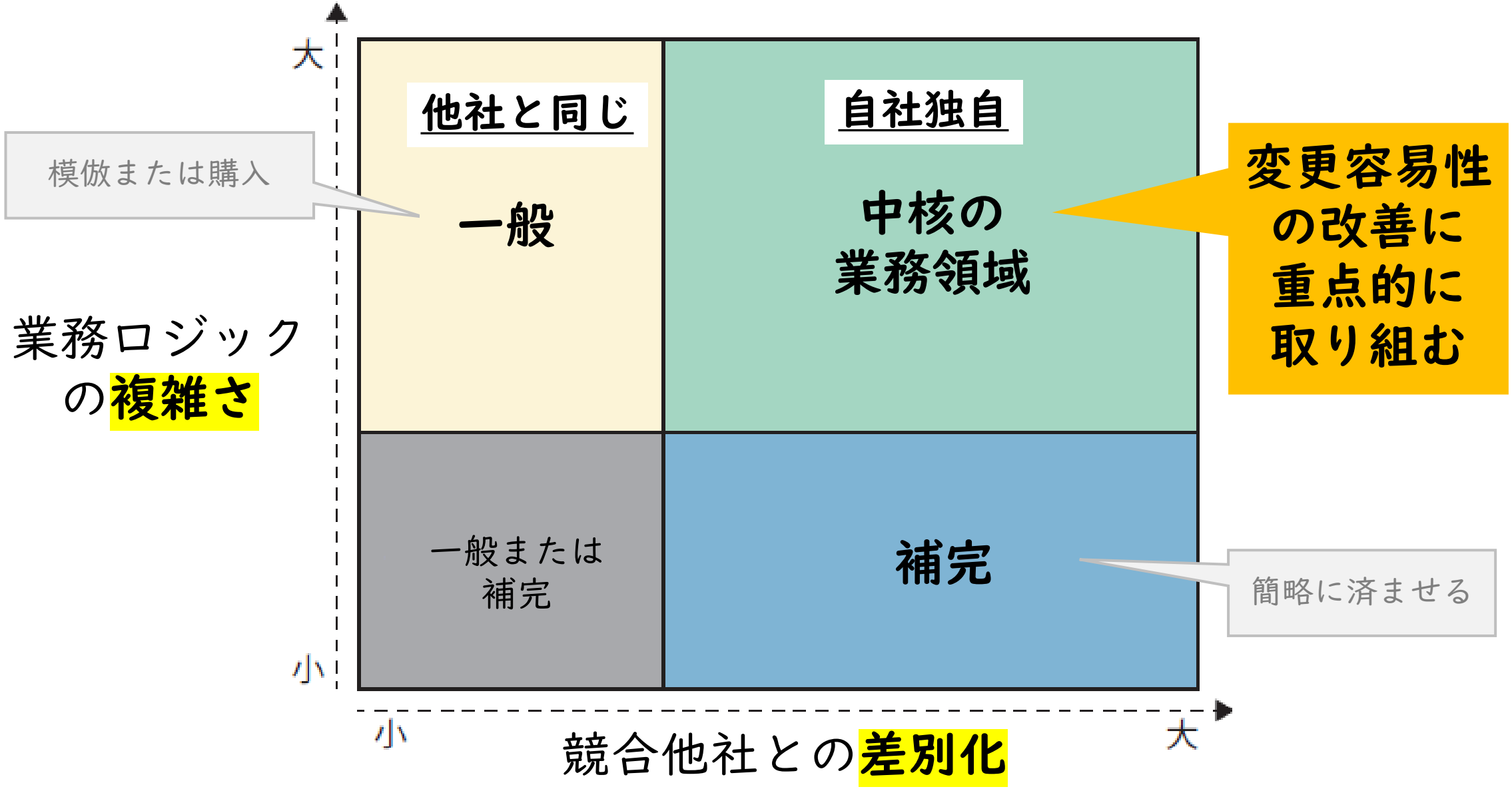
事業戦略とソフトウェアの実装を結びつける



競合他社と異なる
自社独自の価値提案

差別化戦略の実行手段
適切な行動を刺激
不適切な行動を制限

ビジネスルールに基づく
計算判断ロジックの実装


大
小
他社と同じ
自社独自
模倣または購入
一般
中核の
業務領域
変更容易性
の改善に
重点的に
取り組む
業務ロジック
の複雑さ
一般または
補完
補完
簡略に済ませる
小
大
競合他社との差別化

戦略的な設計改善の参考図書

【エッセンシャル版】

まとめ

① 事業活動とソフトウェアシステムは、広く深く連動して、双方向で作用しながら発展を続ける

② ソフトウェアの変更容易性が事業価値を生む

③ 整理整頓されたコードは乱雑なコードより変更が楽で安全

④ 事業戦略とソフトウェアの実装を結びつける

（参考）
戦略的データマネージメントの実践技法

データ管理の基本原則

記録と利用を別の設計にする

事実の記録の完全性を追求する

データ利用の多様性に柔軟に対応する

データの記録と利用を別の設計にする

記録の完全性と利用の柔軟性は衝突する

事実の記録の完全性を目指したテーブル設計は、
データ利用の視点からは最適なテーブル設計ではない

記録の関心事と利用の関心事を切り離すことで、
それぞれの関心事に最適な設計を選択しやすくなる

事実の記録の完全性

イミュータブル

記録した事実を変更しない

NOT NULL

NULL（不定）という事実はない

記録の同時性

異なるタイミングで発生するデータは別テーブルに分ける

自由度の制限

可能な限り範囲が狭いデータ型で記録する

有効なデータを、参照制約とチェック制約で制限する

記録の完全性を改善する

- **戦略的に重要な事実の欠落を検知する**
  - 記録可能だが、記録されていないデータがないか？
  - 記録しているが、上書きして消失しているデータがないか？
  - 記録しているが、データ型やデータ範囲の制約に問題はないか？

- **戦略的に重要な事実を完全に記録できるように、アプリケーションとデータベースを設計改善する**

利用の多様性に対応する工夫

- **第１選択肢：事実の記録から動的に導出する**
- **第２選択肢：事実を記録するタイミングで、参照用のデータを導出して書き込む（同期、非同期）**
- **記録するデータ空間と、参照するデータ空間は厳密に隔離する**
  最低限、論理的に分ける、できれば物理的に分ける

利用の多様性に適切に対応する

- 性能面の問題は、計測してから改善に取り組む
- あるユースケースに特化したテーブルを他の用途に利用しない
- ユースケースが不明確な汎用的な参照テーブルを作成しない