uVALUE

# Contents

## SOA実現のための
## コンポーネントベースモデリングの実例

1. SOAにおけるコンポーネント
2. コンポーネントベースモデリングのアプローチ
3. 業務分析・要求分析・システム分析の事例
4. コンポーネント化推進における参考事例

株式会社 日立製作所　ソフトウェア事業部

HITACHI
Inspire the Next

uVALUE

# Contents

## SOA実現のための
## コンポーネントベースモデリングの実例

1. SOAにおけるコンポーネント
2. コンポーネントベースモデリングのアプローチ
3. 業務分析・要求分析・システム分析の事例
4. コンポーネント化推進における参考事例

HITACHI
Inspire the Next

# 1-1 SOAとは

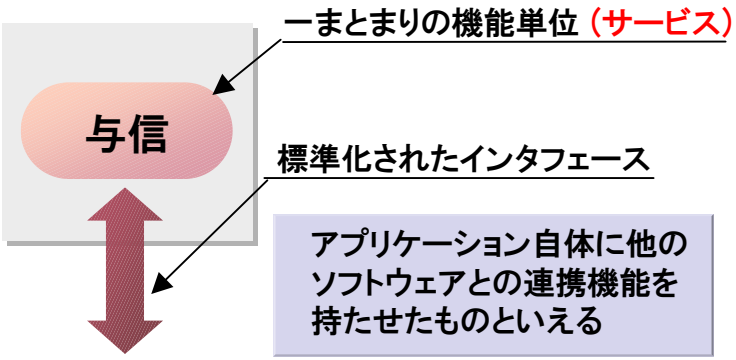
**ポイント**

- SOA 技術とは、「サービスの組合せ」でシステムを構築すること
  - アーキテクチャとして（分散処理、業務の部品化）
  - アプローチとして　　（業務プロセスの可視化、サービスの再利用）

■サービスとは？

・外部から**標準化された手順**によって呼び出せる**一連のソフトウェア群（＝コンポーネント）**

・**単体で**人間にとって**意味のある単位の機能**を有するソフトウェア群

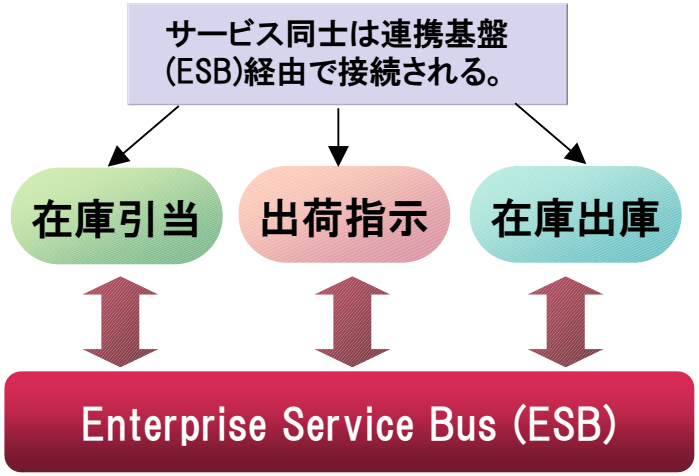



■組合せとは？

・業務に合せて**必要なサービスを緩やかに結合**することで柔軟なシステムを構築

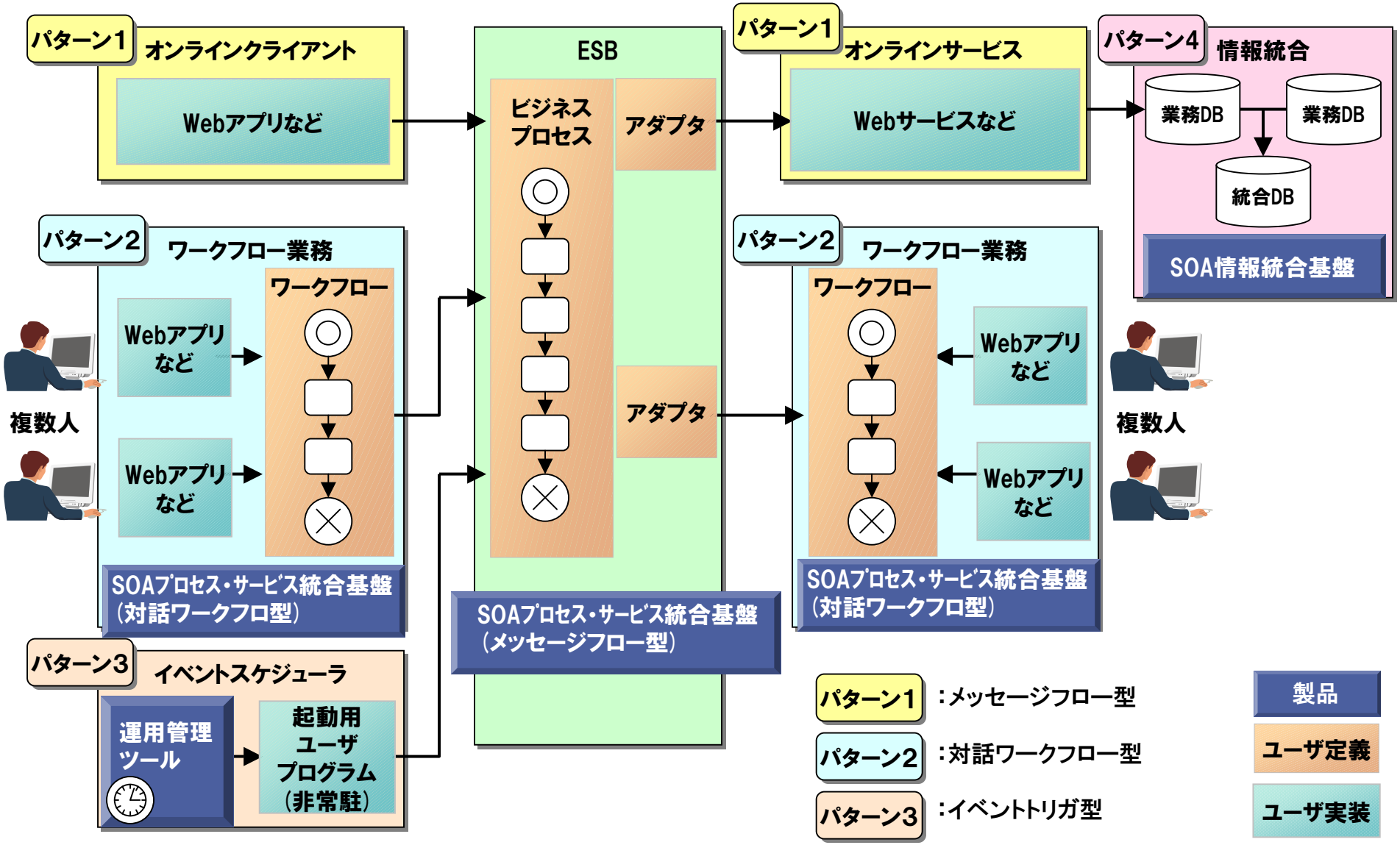
・業務をサービスの集合としてとらえることで、**プロセスの可視化**を実現

・サービスの**部品化**を進めることで、**再利用**を促進




「サービス」として分解された一連の業務処理を組合せて、業務プロセス全体を構築する考え方がSOA

# 1-2 SOAの形態

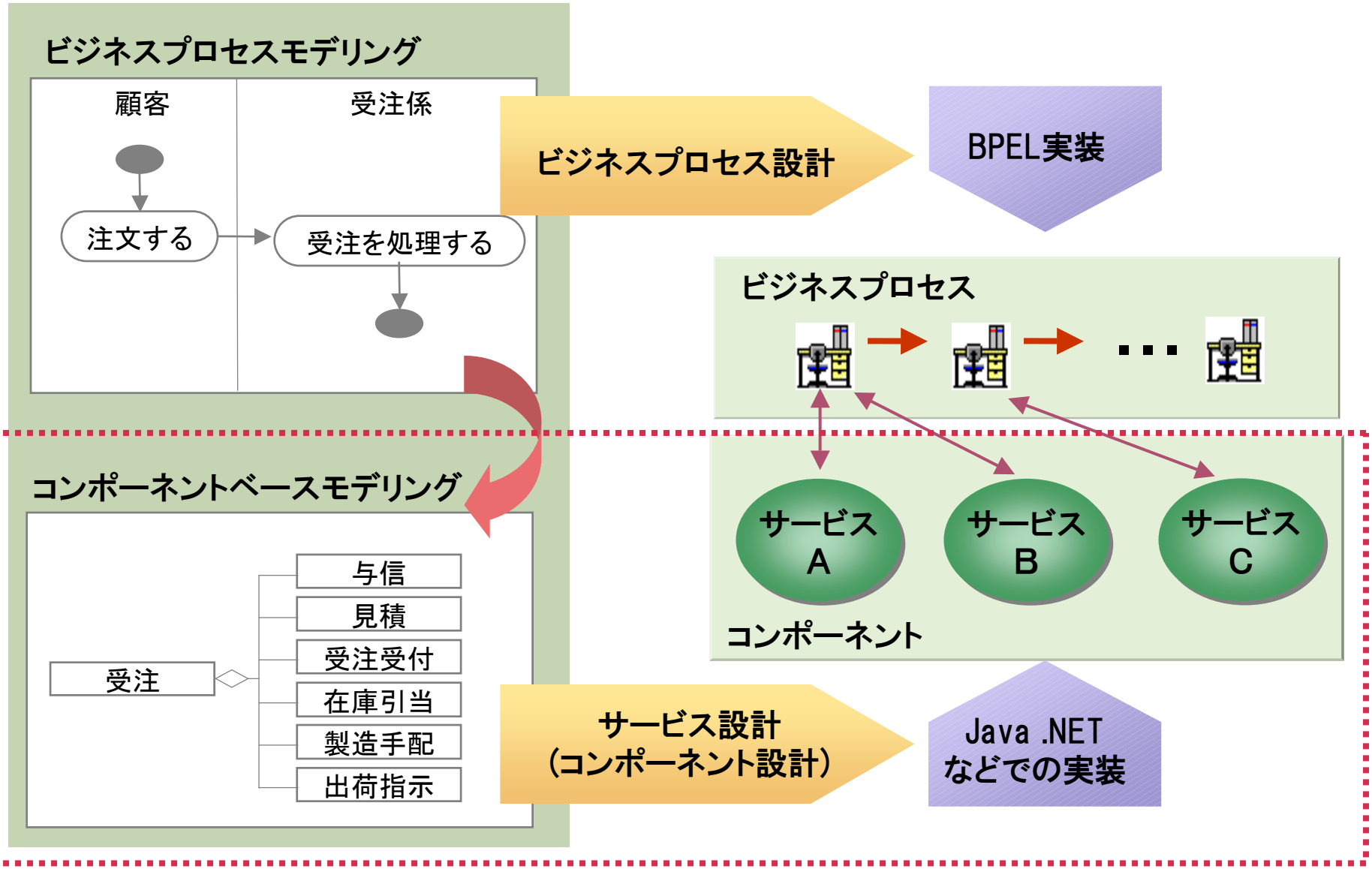


# 1-3 SOAの適用パターン

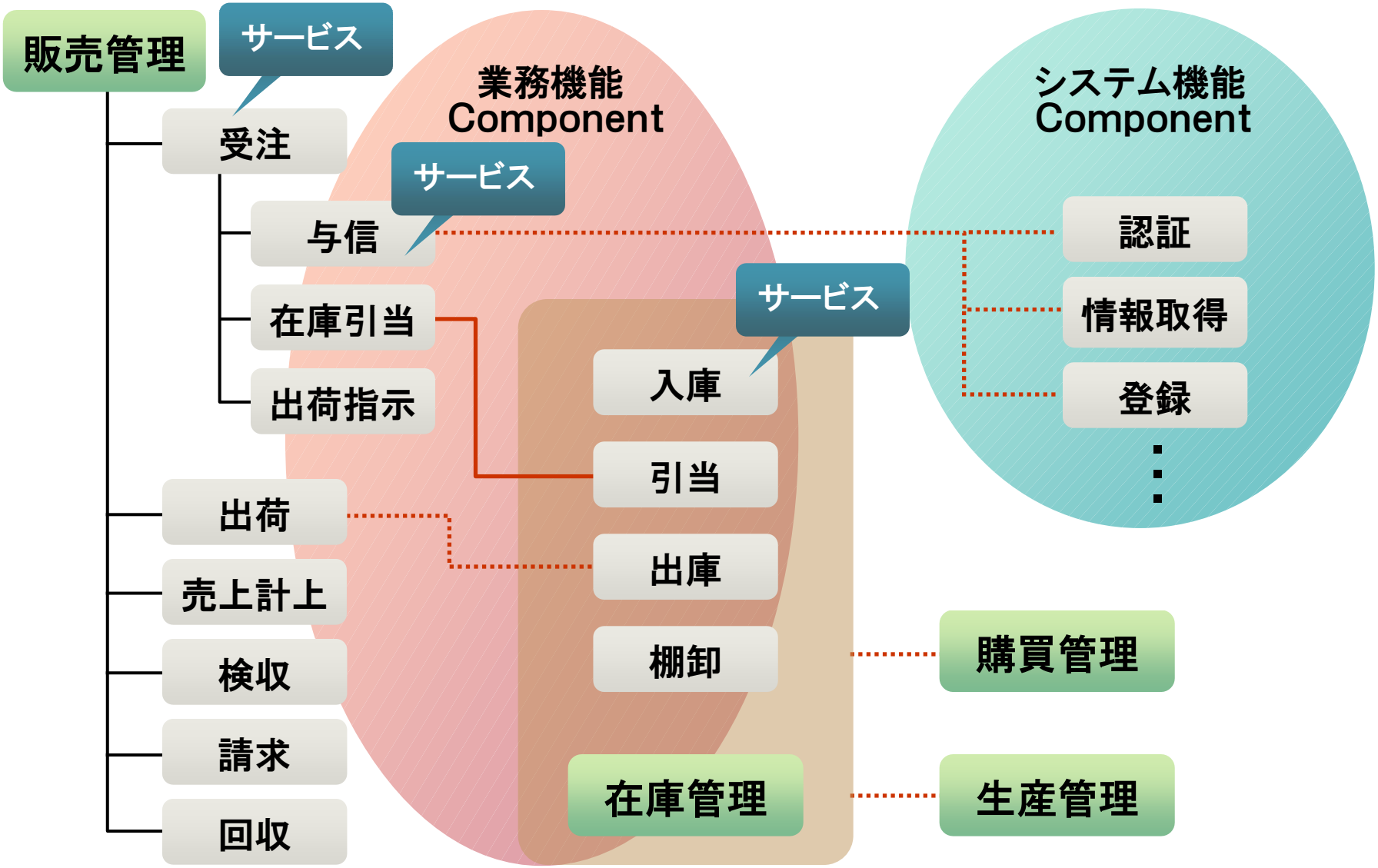
| パターン | 概要 | 適用 |
|---|---|---|
| メッセージフロー型 | ビジネスプロセスに沿って、各サービスが同期、非同期で処理実行されていくような形態 | 在庫の引当(サービス)→出荷指示(サービス)のように、一連のサービスがビジネスプロセスで接続されるような部分 |
| 対話ワークフロー型 | ワークフローの定義に沿って、同一サービスが対話型で実行されるような形態 | 受注受付(サービス)において、入力、審査、承認など、同一サービスが対話型のフローで処理されるような部分 |
| イベントトリガ型 | 時刻指定など、バッチ処理的に外部から起動がかかるような形態 | バッチ処理でのデータ抽出～各システムへの自動配布など、特定時刻で起動され、かつ一連の流れがシナリオ化できるような部分 |
| 情報統合型 | 情報そのものが統合されるような形態 | Master of Masterや集合体マスタの作成のように、マスタの連携や統合を図る必要がある部分 |

上記パターンの組合せも存在する

# 1-4 SOAの設計形態



# 1-5 コンポーネントの粒度



# 1-6 SOAにおけるサービスとコンポーネントの関係

## システム機能コンポーネント

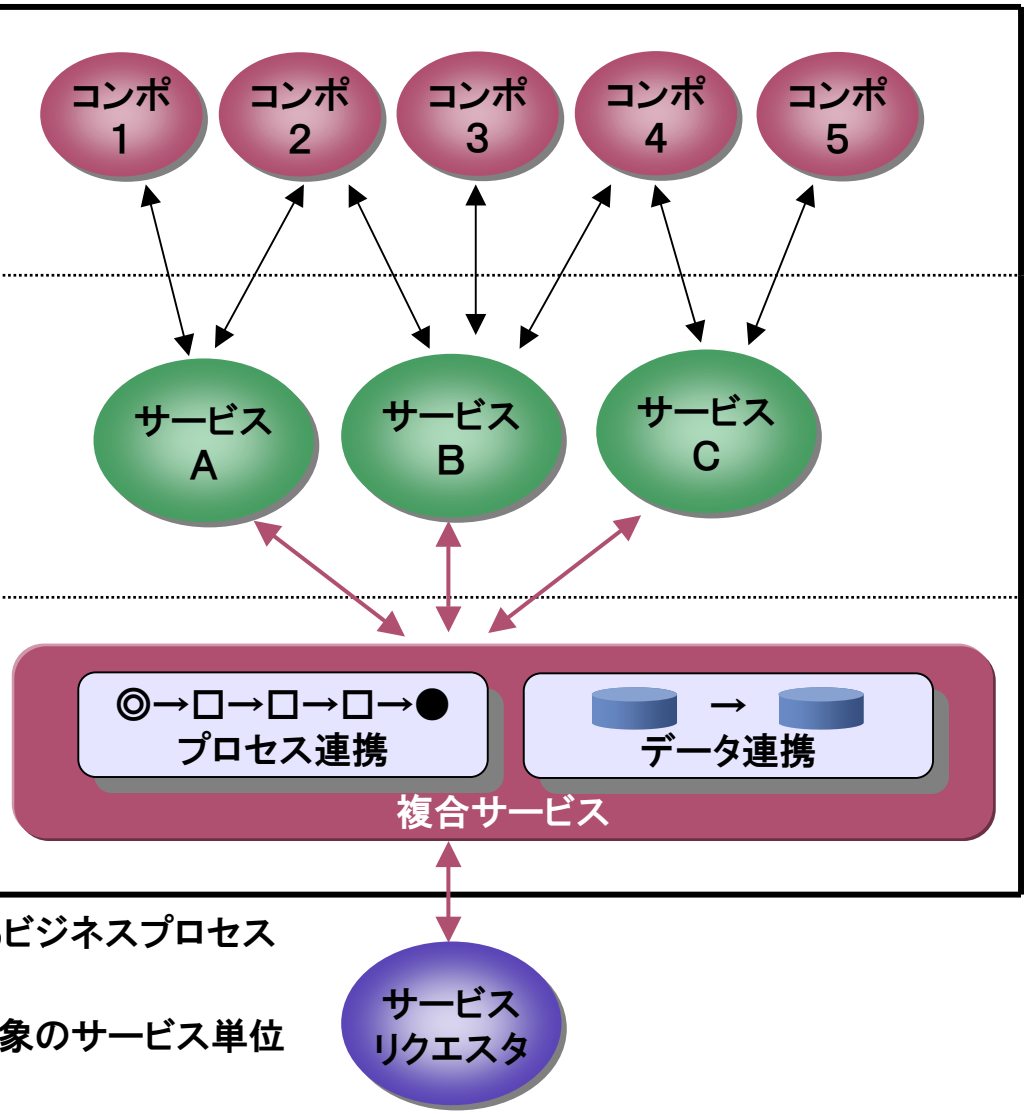
AP開発のための共通部品
例：案件登録/更新/削除、製品情報検索、
共通計算処理、帳票出力　etc

## 業務機能コンポーネント

業務処理を構成する最小単位
例：見積依頼受付、見積回答作成、見積
書出力、etc

## ビジネスコンポーネント

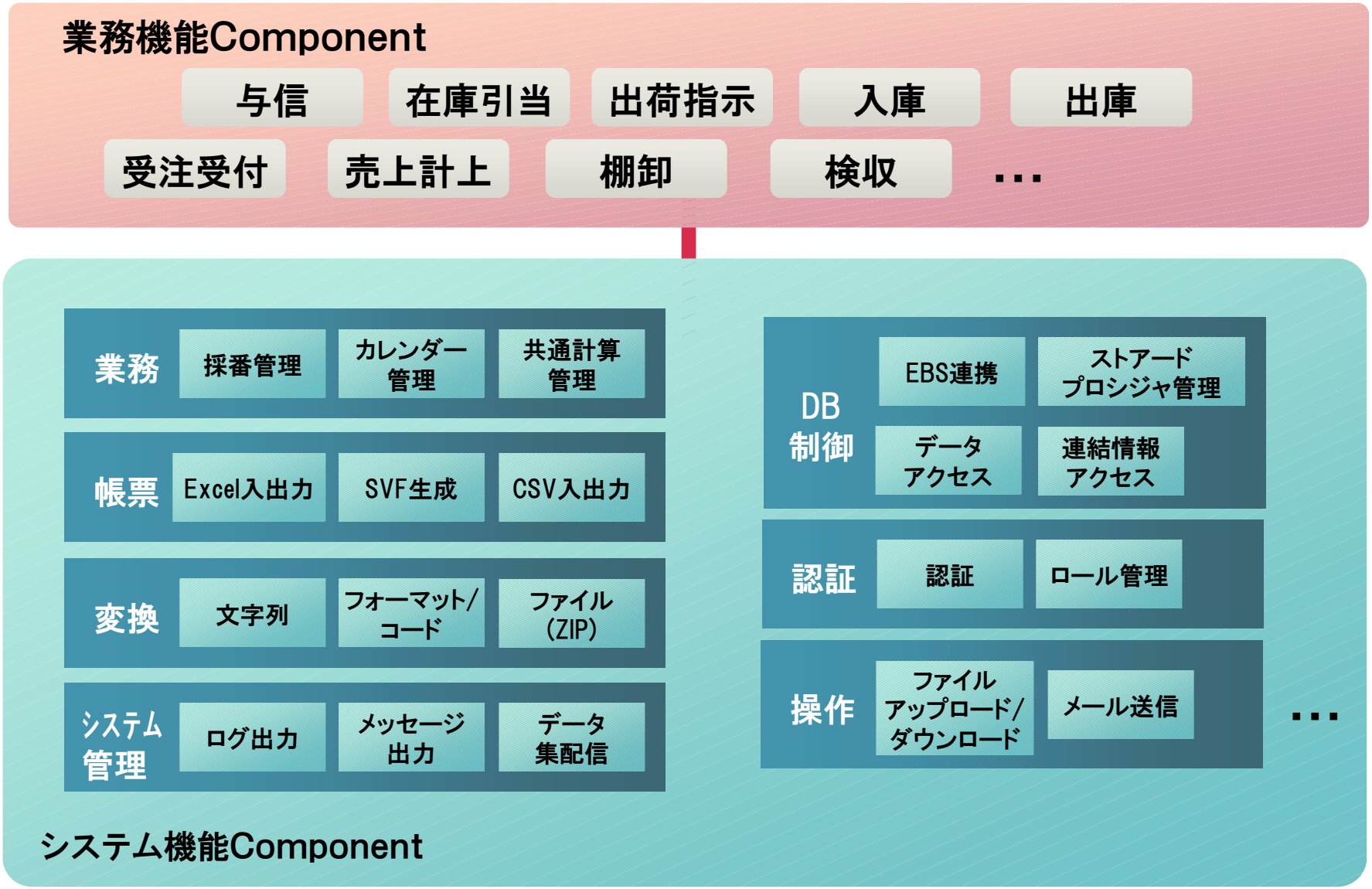
一つのまとまった業務処理
例：見積り、受注、与信、輸出管理、etc



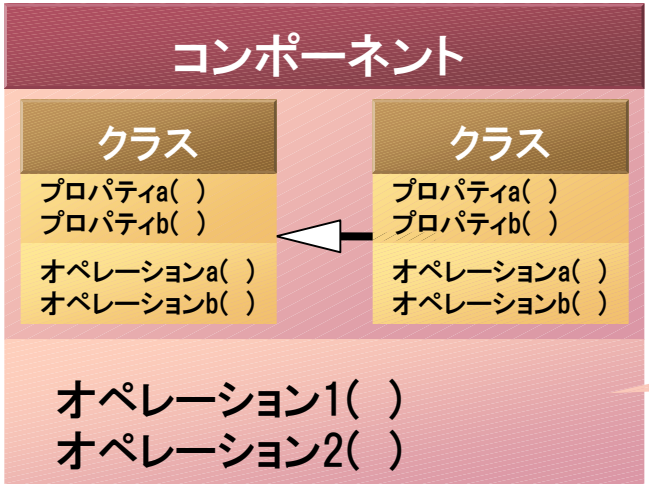

※　複合サービスの幾つかを更に連携させるビジネスプロセスも存在する

※　レガシーAPも一つのサービスであり、対象のサービス単位はそのAPのI/Fと考えることが出来る

# 1-7 業務機能とシステム機能コンポーネントの事例



# 1-8 コンポーネントのパターン




- コンポーネントを構成する最小単位

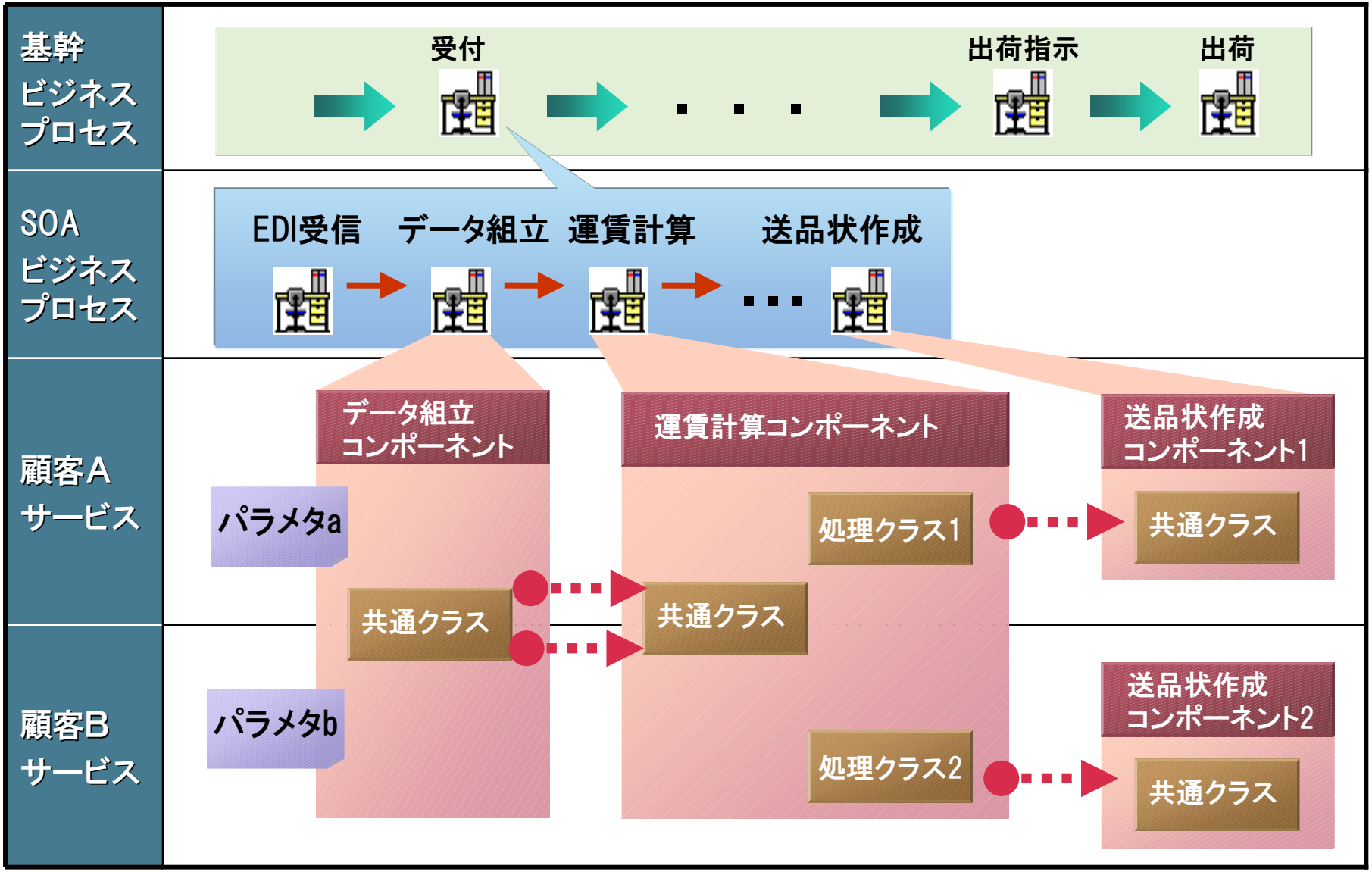
  呼び出しクラスがオペーレーションとパラメタ(変数)を指定してクラスを呼び出す

- コンポーネントに対するインタフェース

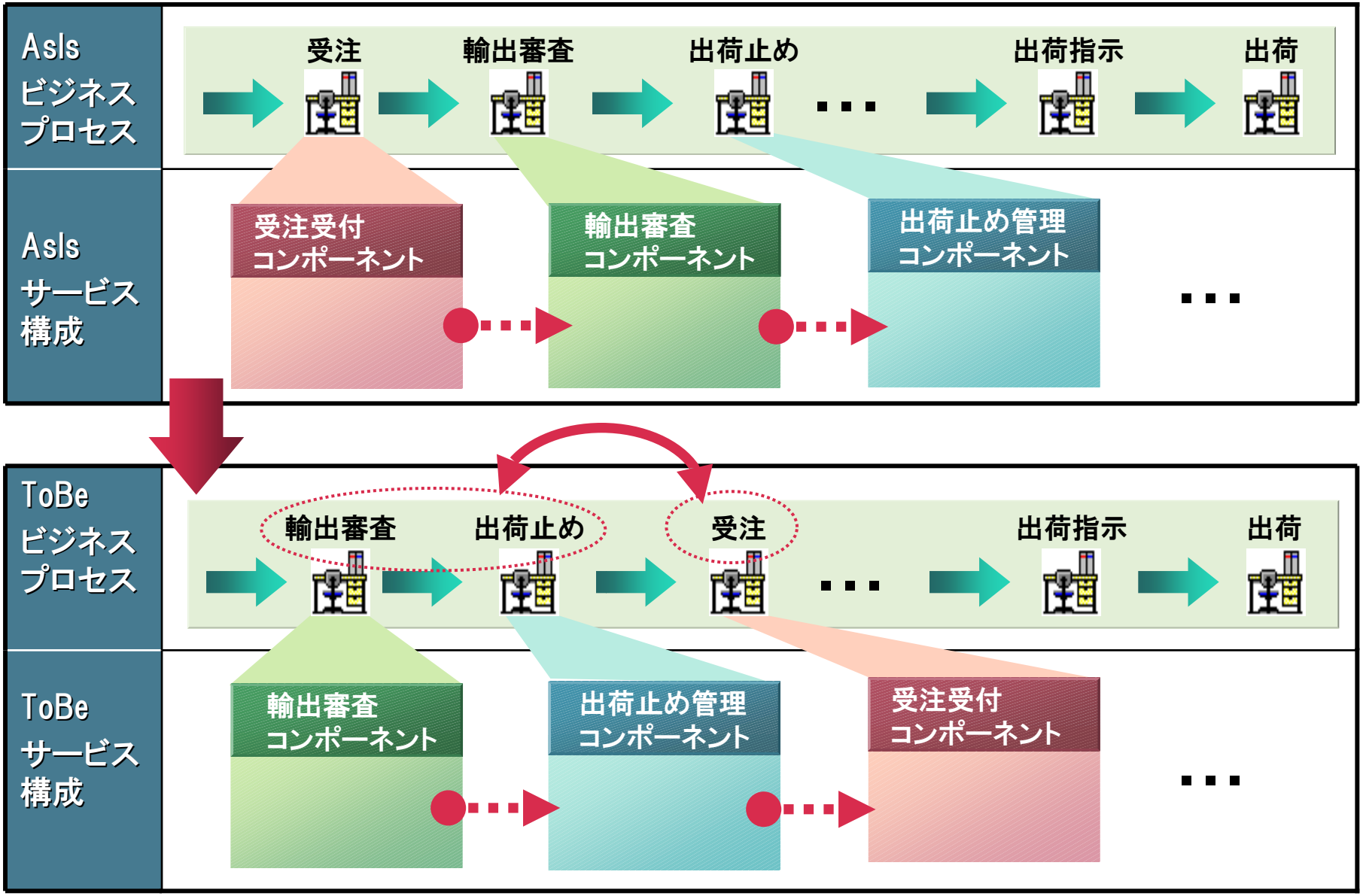
  呼び出し元がオペーレーションとパラメタ(変数)を指定してコンポーネントを呼び出す

| コンポーネントパターン | 概　　要 | 適用業務処理 |
|---|---|---|
| 完全共有型 | コンポーネントの中のクラスも全て共有し、パラメタや呼び出しのオペレーションよって処理内容を変えることで、コンポーネントを恒久的に共有する形態 | フォーマット変換、データアクセス etc |
| クラス継承による共有 | 一つのコンポーネントの中で共通クラスとは別に、異なる処理クラスを設け、パラメタまたはオペレーションの呼び出しにより、共通クラスを継承して異なる処理クラスを実行させる形態 | 在庫引当における引当処理とその引当条件　etc |
| 別コンポーネント | 共通部分がまったくなく、全てが別処理となるようなケースで、別コンポーネントとして用意する形態 | |

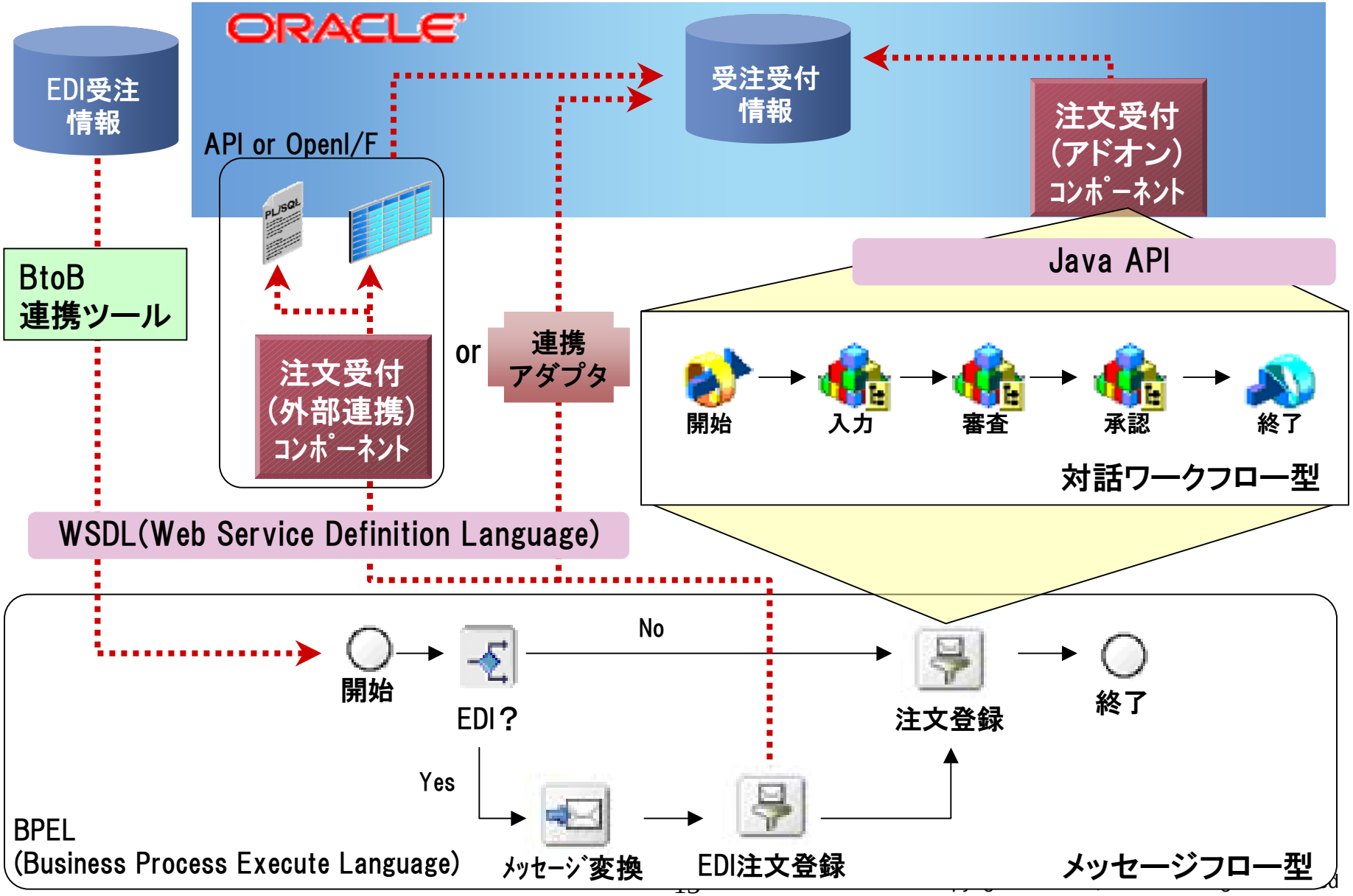
# 1-9 SOAとコンポーネント化の事例（サービス組立）



# 1-10 SOAとコンポーネント化の事例（サービス組替え）



# 1-11 SOAとコンポーネント化の事例（ERP連携）



uVALUE

# Contents

## SOA実現のための
## コンポーネントベースモデリングの実例

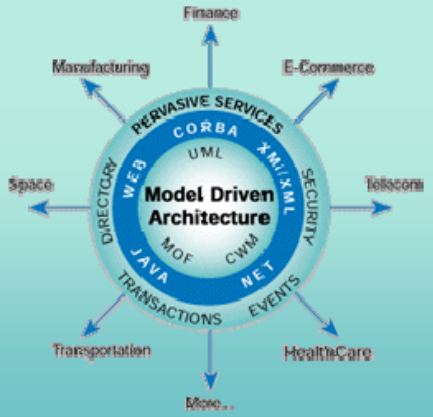
1. SOAにおけるコンポーネント
2. コンポーネントベースモデリングのアプローチ
3. 業務分析・要求分析・システム分析の事例
4. コンポーネント化推進における参考事例

HITACHI
Inspire the Next

# 2-1 MDAとUML

## MDA(Model Driven Architecture):
## モデル駆動型アーキテクチャ

- モデリング主導のシステム開発を実現する参照アーキテクチャ
- OMG* （国際標準化団体)が普及活動中




## UML(Unified Modeling Language):
## 統一モデリング言語

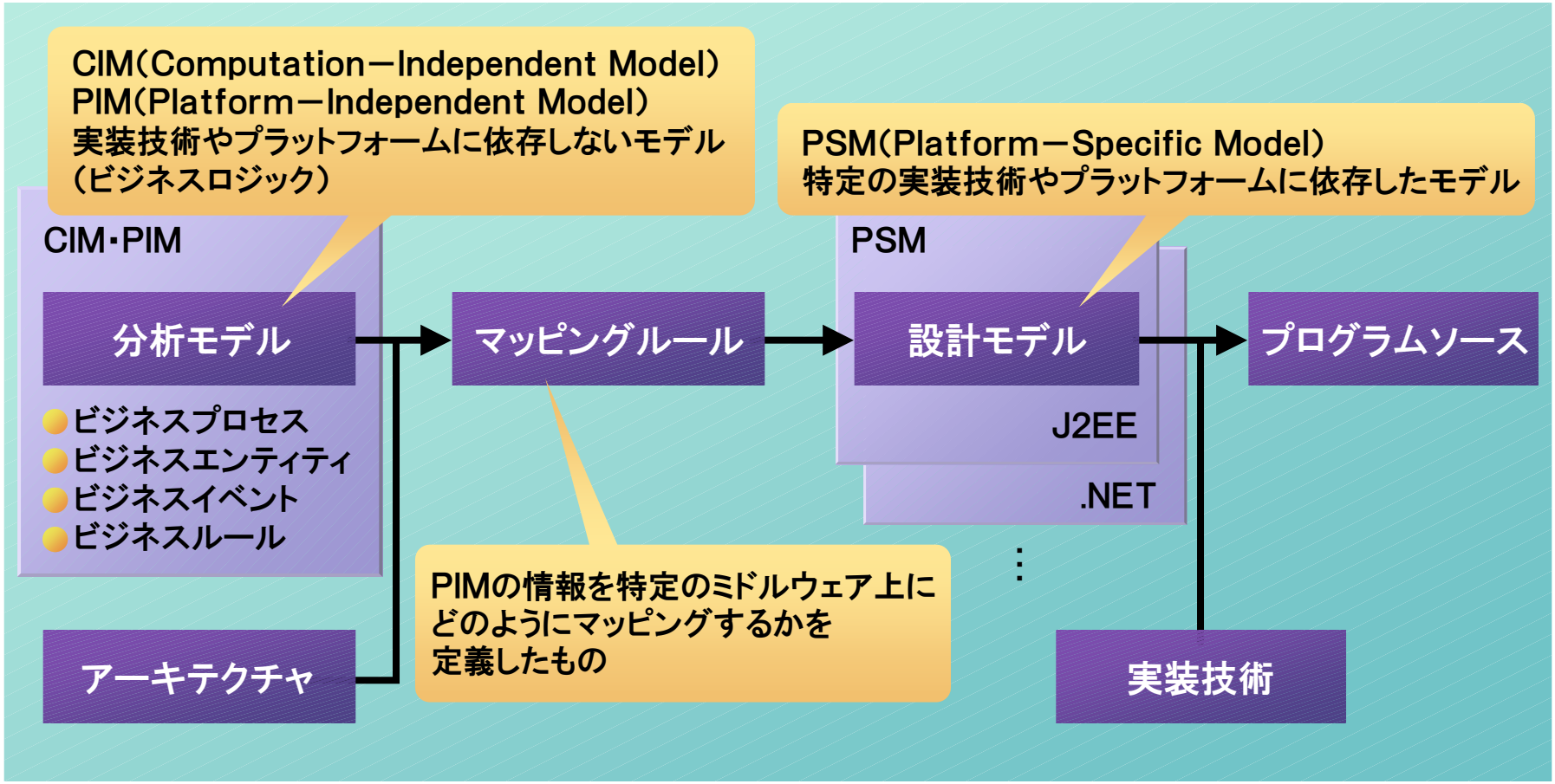
- 分析／モデリングの際に利用する「言語」
- 1997年にOMGが標準化


＊OMG(Object Management Group):
オブジェクト指向技術の標準化、普及をすすめるため、1989年に設立された標準化団体

# 2-2 MDAによる開発アプローチ

## MDA(Model Driven Architecture)

業務モデルとITモデルを分離し、それぞれ独立したモデルで
定義し、プラットフォーム決定時、業務モデルを設計モデルにマッピングする体系。



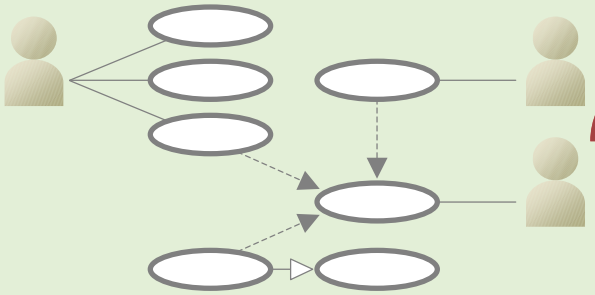
# 2-3 UMLの種類（UML2.0の例）

| UML種別 | 概　　要 |
|---|---|
| アクティビティ図 | 業務フローなどアクションの実行の流れを表現できる |
| ユースケース図 | アクター(システム外のユーザや他のシステムなど)との間での外部アクションとその関係を表現できる |
| クラス図 | クラス(型)が持つ属性と、その操作の関係を表現できる |
| オブジェクト図 | クラス図を導き出すための補助的なものであり、クラス図と異なるのはインスタンス(クラスからの生成物)を表現する |
| ステートマシン図 | 特にクラスの中で状態がある場合、その状態遷移を表現する |
| シーケンス図 | 複数のオブジェクト間のメッセージの流れを時間軸中心で表現できる |
| パッケージ図 | クラスなどをグルーピングして表現できる |
| 複合構造図 | 全体ー部分構造をもつクラスの実行時の構造を示す |
| コミュニケーション図 | インスタンス間の相互作用を構造中心に表現する |
| インタラクション・オーバービュー図 | 条件によって異なる動作をするシーケンス図を、アクティビティー図の中に含めることで表現する |
| タイミング図 | インスタンス間の状態遷移や相互作用を時間制約付きで表現する |
| コンポーネント図 | 実装コンポーネントとその関係を表現できる |
| 配置図 | ハードウェアとその間の関係を表現できる |

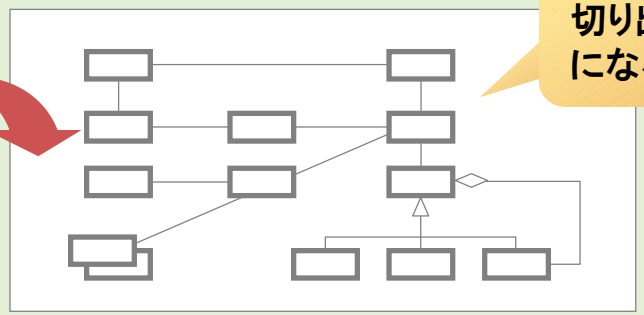
# 2-4 コンポーネントベースモデリングの特徴

## 一般的な手法

膨大なユースケース



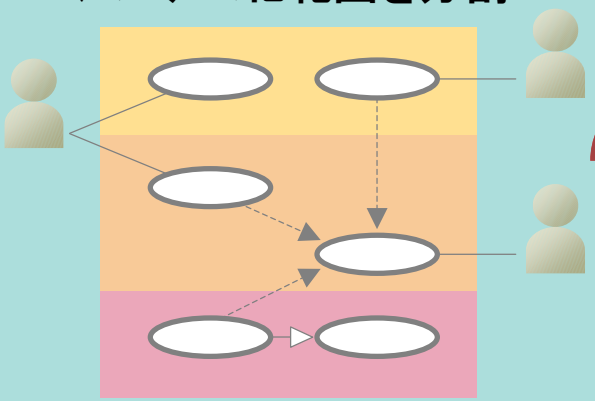
膨大なクラスと複雑な関連



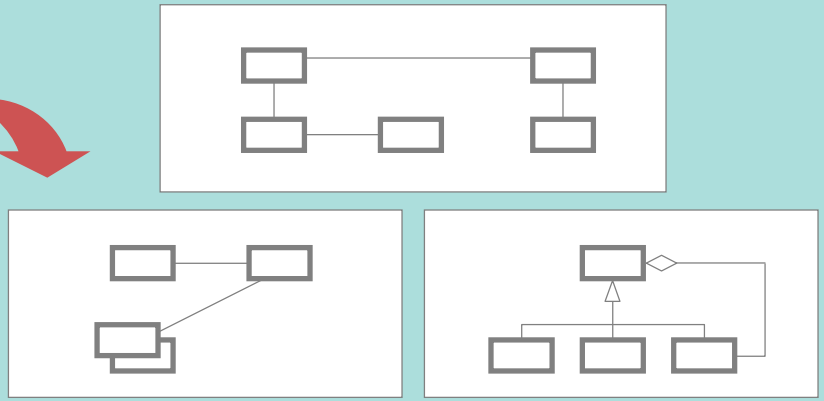
コンポーネントの切り出しが困難になる

## コンポーネントベース

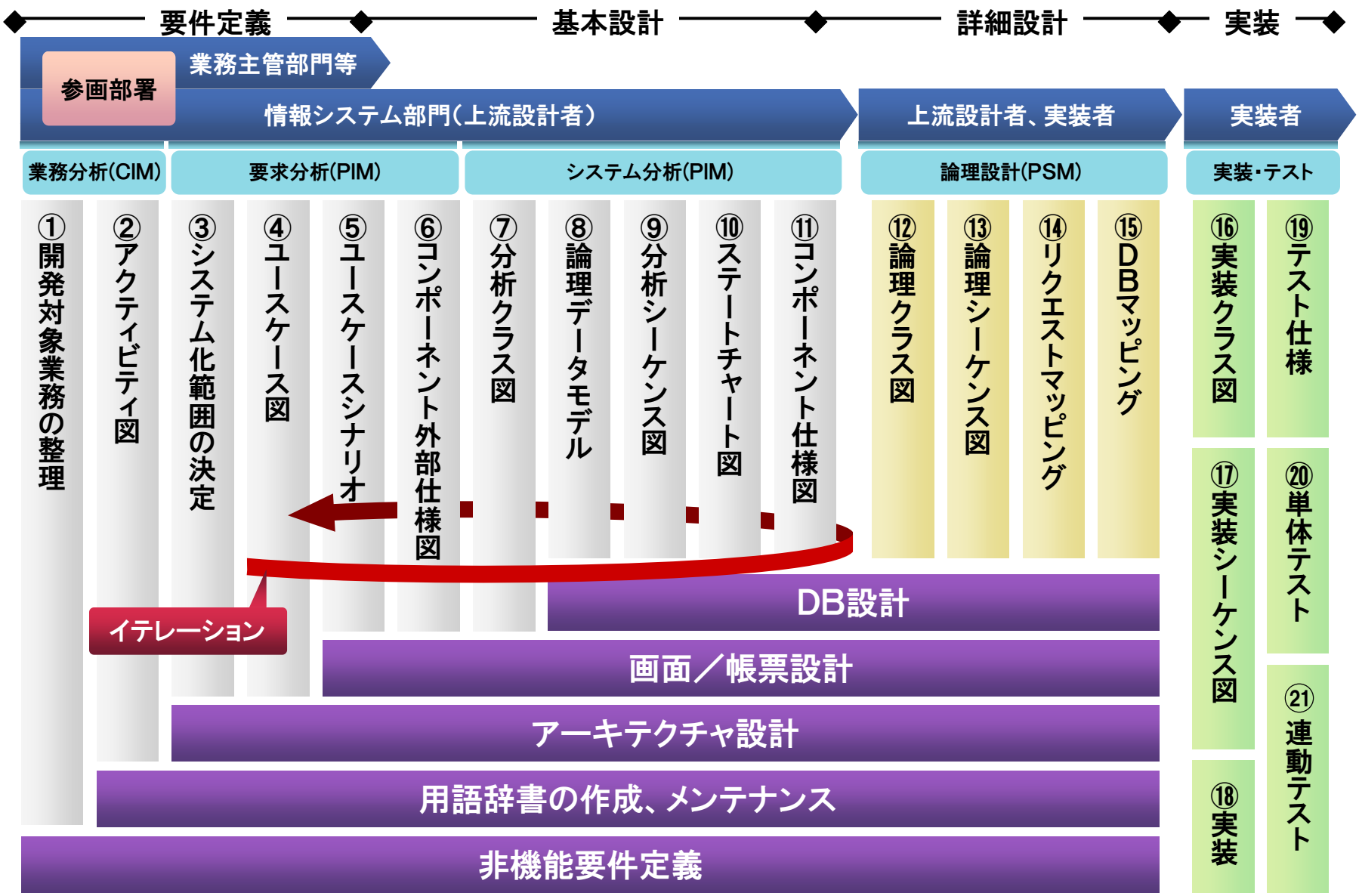
モデリングの段階から
システム化範囲を分割

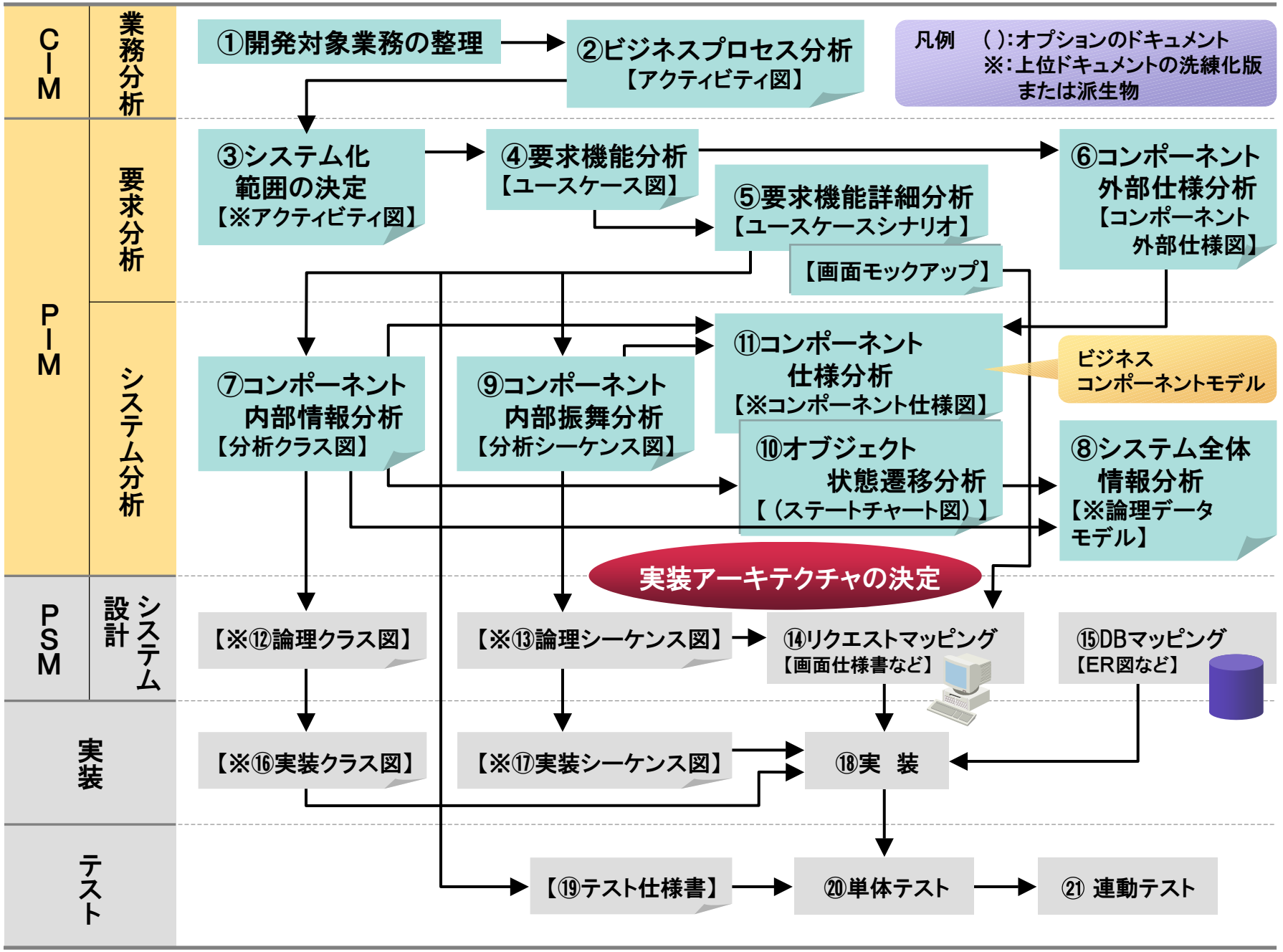


複数のシンプルなクラス図



# 2-5 コンポーネントベース開発アプローチ




CIM
業務分析
PIM
要求分析
システム分析
PSM
システム設計
実装
テスト
①開発対象業務の整理
②ビジネスプロセス分析
【アクティビティ図】
凡例 ( ):オプションのドキュメント
※:上位ドキュメントの洗練化版
または派生物
③システム化
範囲の決定
【※アクティビティ図】
④要求機能分析
【ユースケース図】
⑤要求機能詳細分析
【ユースケースシナリオ】
【画面モックアップ】
⑥コンポーネント
外部仕様分析
【コンポーネント
外部仕様図】
⑦コンポーネント
内部情報分析
【分析クラス図】
⑨コンポーネント
内部振舞分析
【分析シーケンス図】
⑪コンポーネント
仕様分析
【※コンポーネント仕様図】
ビジネス
コンポーネントモデル
⑩オブジェクト
状態遷移分析
【(ステートチャート図)】
⑧システム全体
情報分析
【※論理データ
モデル】
実装アーキテクチャの決定
【※⑫論理クラス図】
【※⑬論理シーケンス図】
⑭リクエストマッピング
【画面仕様書など】
⑮DBマッピング
【ER図など】
【※⑯実装クラス図】
【※⑰実装シーケンス図】
⑱実 装
【⑲テスト仕様書】
⑳単体テスト
㉑ 連動テスト

uVALUE

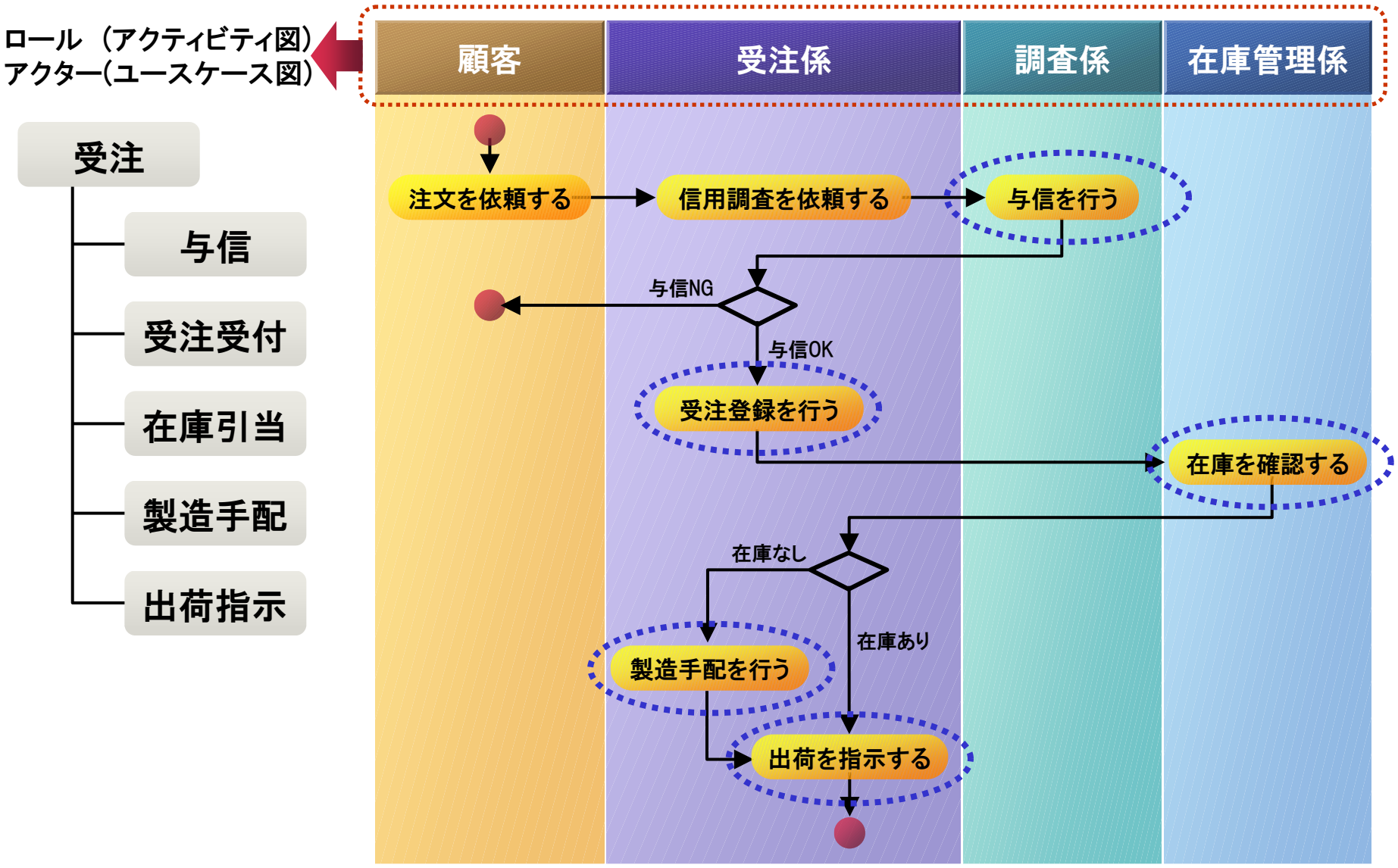
# Contents

## SOA実現のための
## コンポーネントベースモデリングの実例

1. SOAにおけるコンポーネント
2. コンポーネントベースモデリングのアプローチ
3. 業務分析・要求分析・システム分析の事例
4. コンポーネント化推進における参考事例

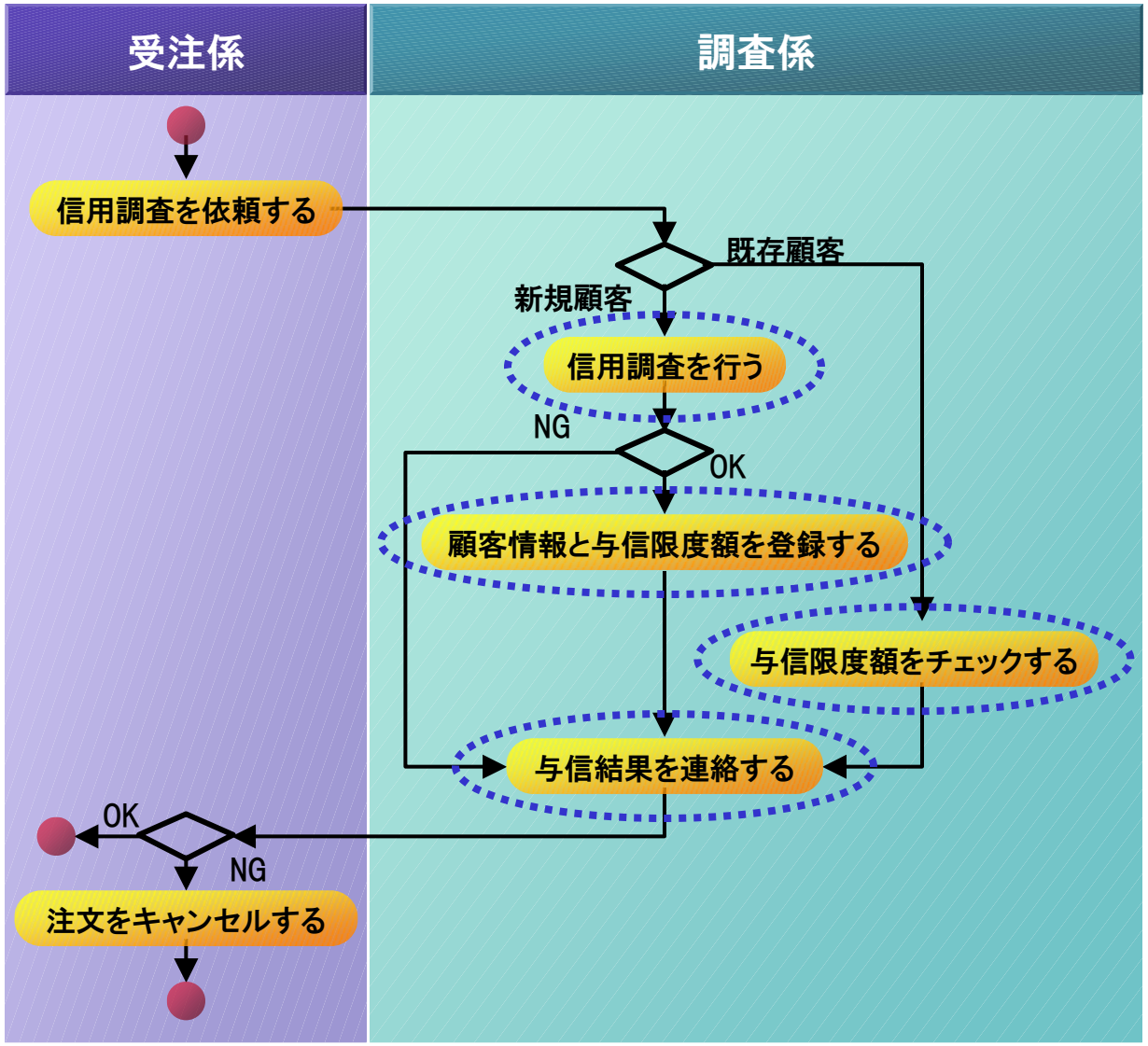
HITACHI
Inspire the Next

# 3-1 開発対象業務の整理とアクティビティ図

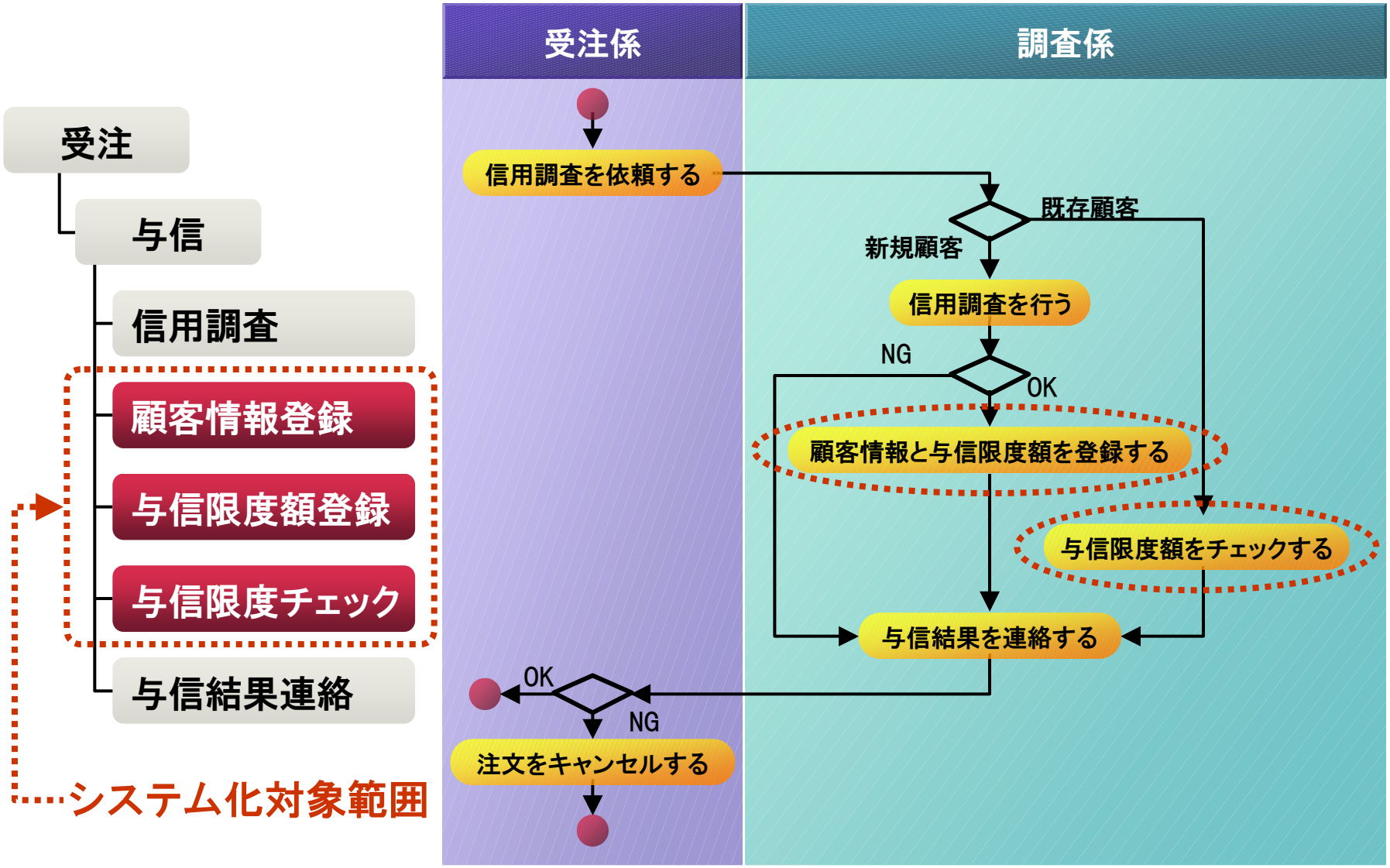


# 3-2 開発対象業務の整理とアクティビティ図（階層）

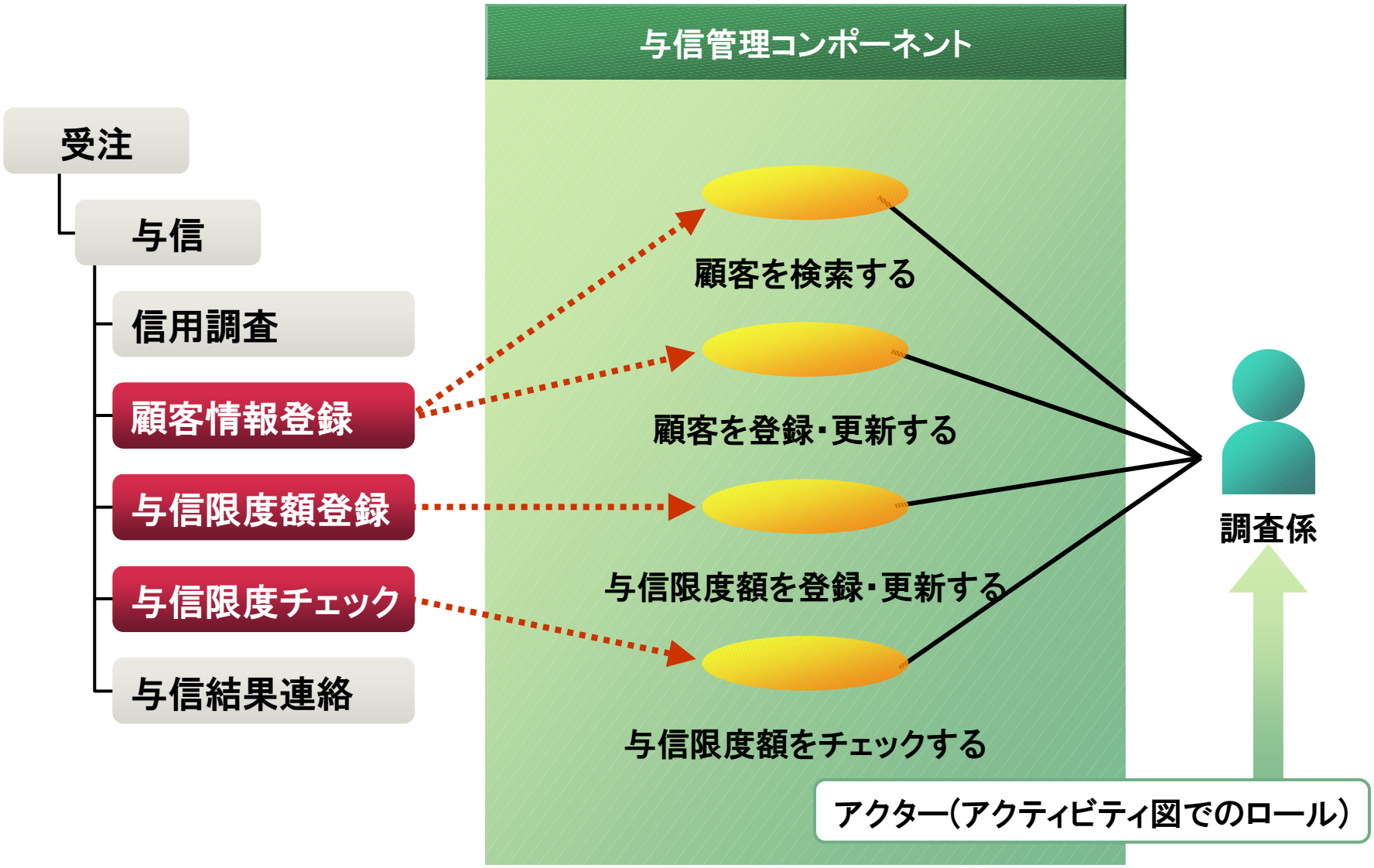
- 受注
  - 与信
    - 信用調査
    - 顧客情報登録
    - 与信限度額登録
    - 与信限度チェック
    - 与信結果連絡



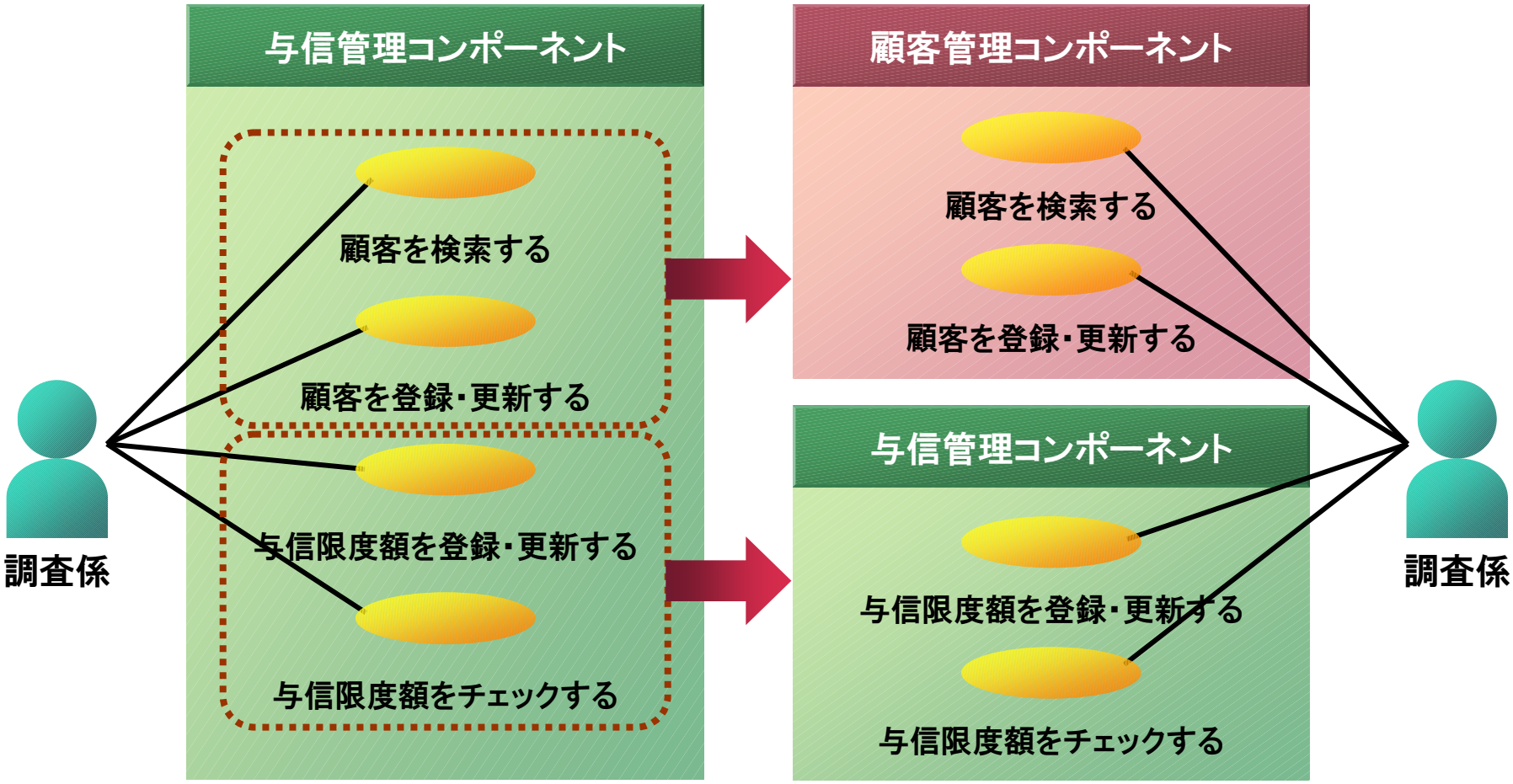
# 3-3 システム化対象範囲の決定



# 3-4 ユースケース図



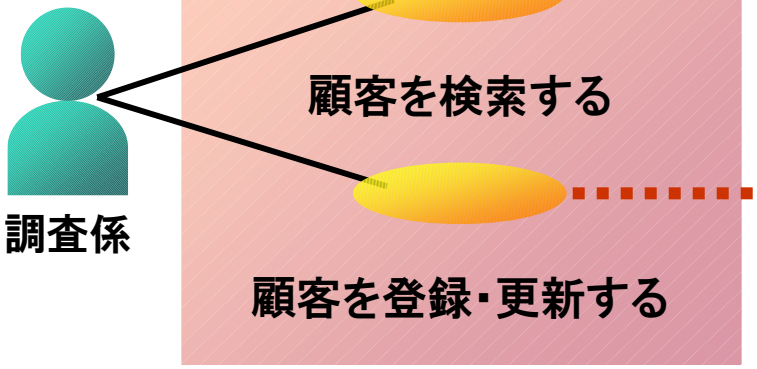
# 3-5 ユースケース図（コンポーネント導出）

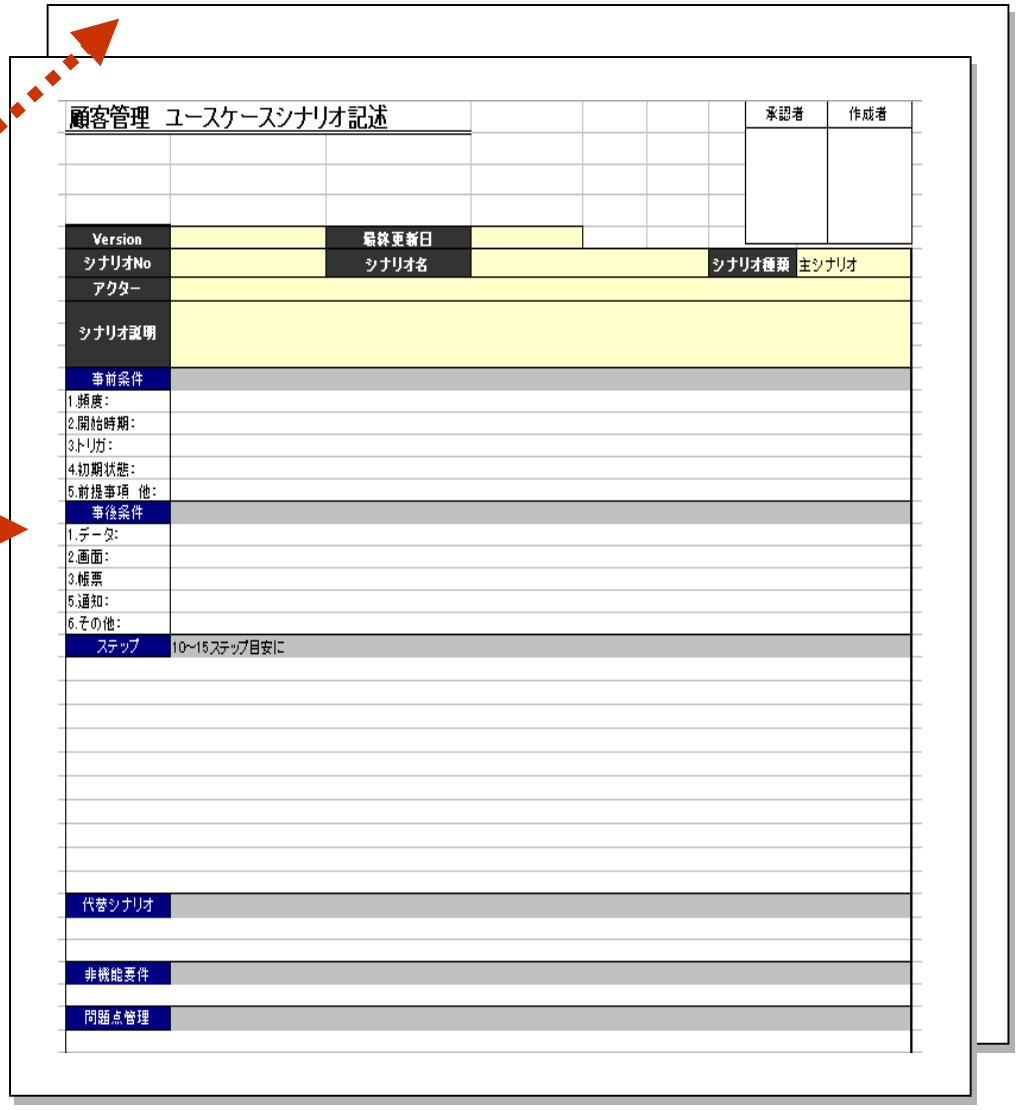



**ユースケースでの共通事項を見出し、必要があればコンポーネント分割を行う**

# 3-6 ユースケースシナリオ

## ユースケース図



顧客管理　ユースケースシナリオ記述

| 承認者 | 作成者 |
|---|---|
| | |

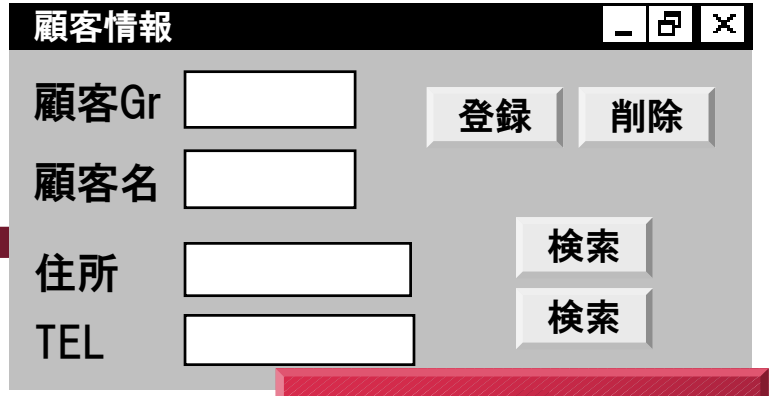
| | | | | | |
|---|---|---|---|---|---|
| Version | | 最終更新日 | | | |
| シナリオNo | | シナリオ名 | | シナリオ種類 | 主シナリオ |
| アクター | | | | | |
| シナリオ説明 | | | | | |
| 事前条件 | | | | | |
| 1.頻度: | | | | | |
| 2.開始時期: | | | | | |
| 3.トリガ: | | | | | |
| 4.初期状態: | | | | | |
| 5.前提事項　他: | | | | | |
| 事後条件 | | | | | |
| 1.データ: | | | | | |
| 2.画面: | | | | | |
| 3.帳票 | | | | | |
| 5.通知: | | | | | |
| 6.その他: | | | | | |
| ステップ | 10~15ステップ目安に | | | | |
| 代替シナリオ | | | | | |
| 非機能要件 | | | | | |
| 問題点管理 | | | | | |

- シナリオはユースケース単位
- シナリオ記述テンプレートなどを用意

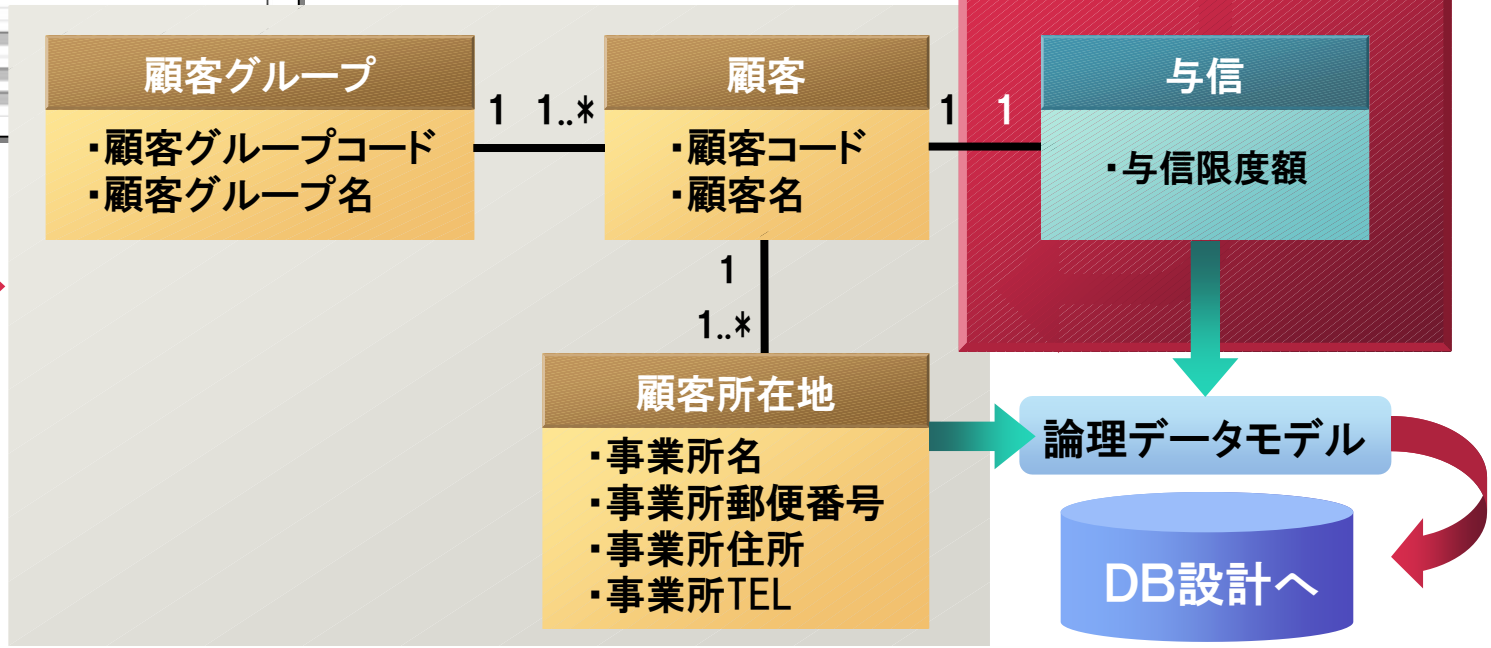
# 3-7 分析クラス図

## ユースケースシナリオ

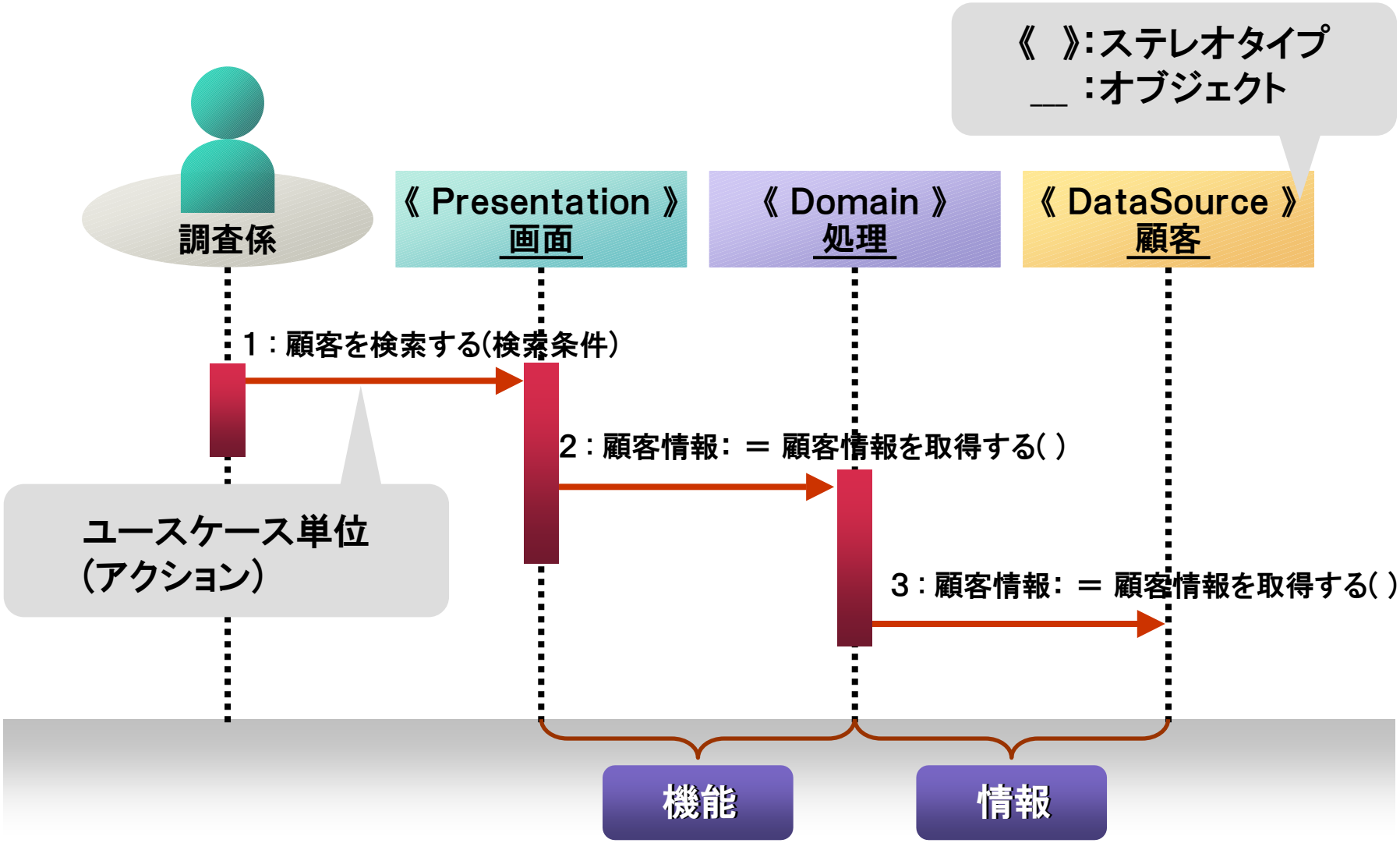

## 画面モックアップ




## 分析クラス図

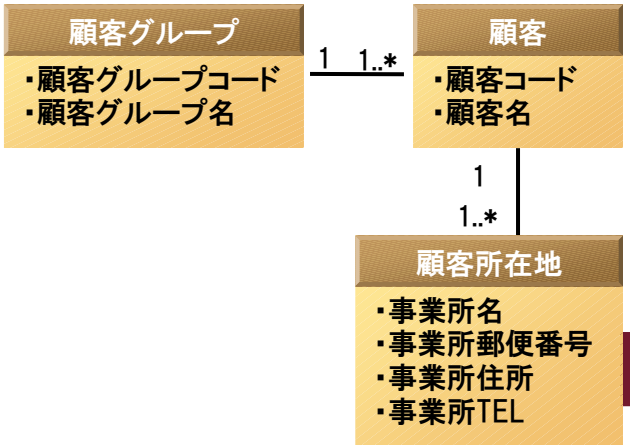


# 3-8 分析シーケンス図

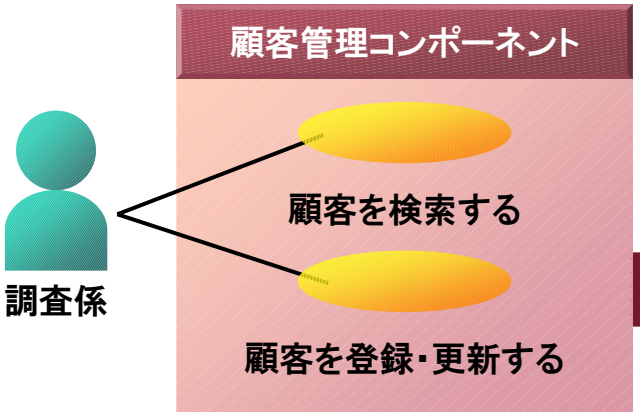


# 3-9 コンポーネント仕様図

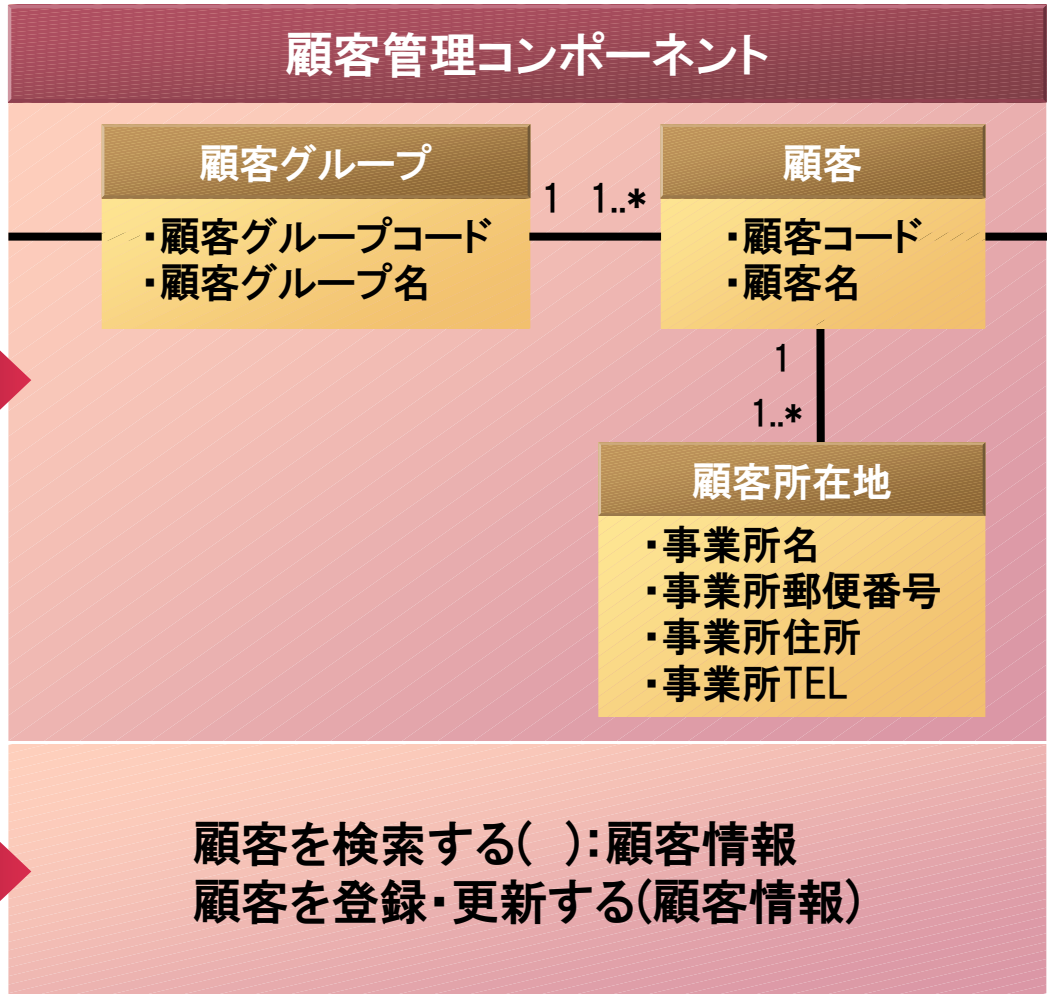
## 分析クラス図(内部仕様)




## ユースケース図(外部仕様)




## コンポーネント仕様



uVALUE

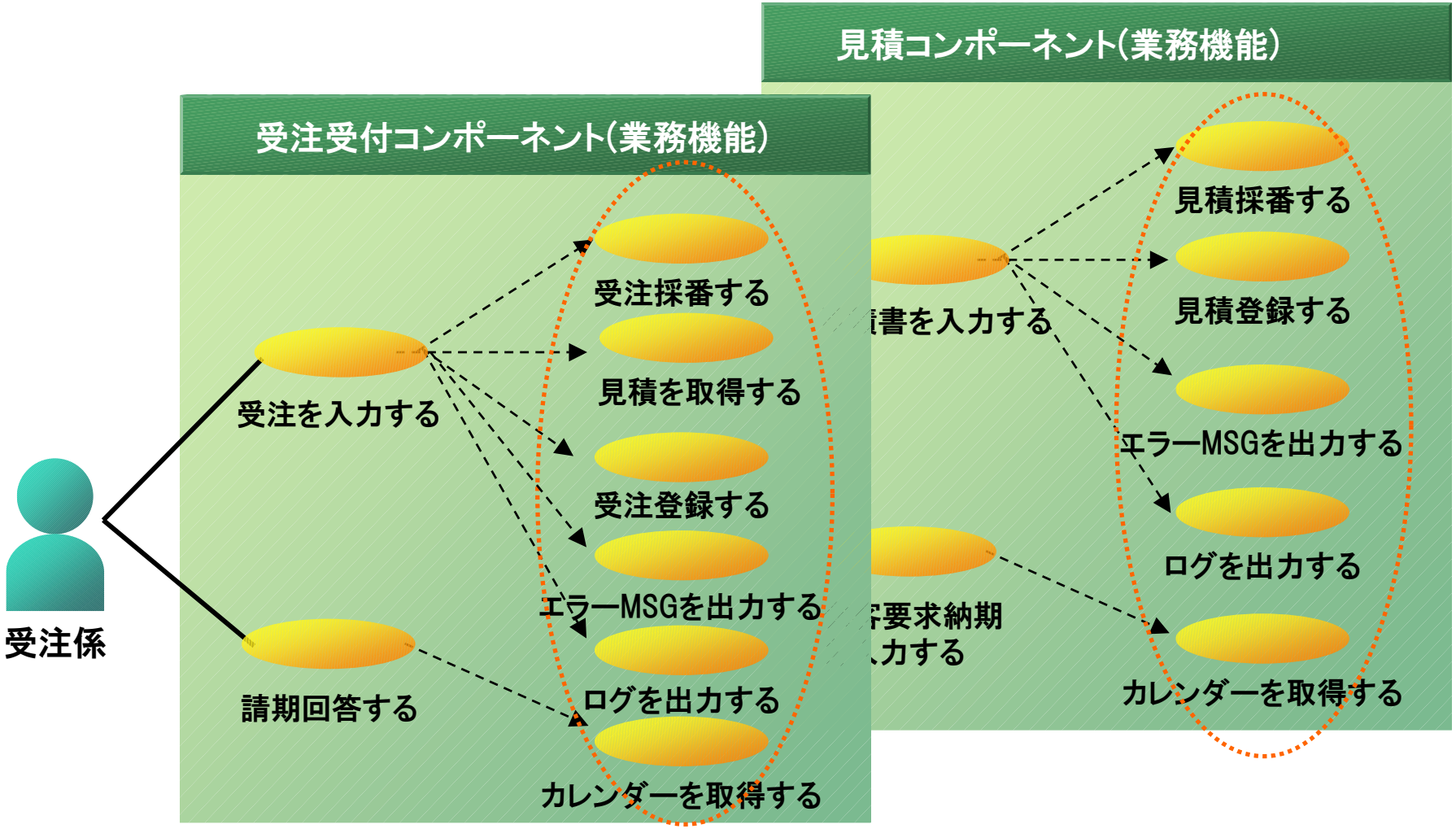
# Contents

## SOA実現のための
## コンポーネントベースモデリングの実例

1. SOAにおけるコンポーネント
2. コンポーネントベースモデリングのアプローチ
3. 業務分析・要求分析・システム分析の事例
4. コンポーネント化推進における参考事例

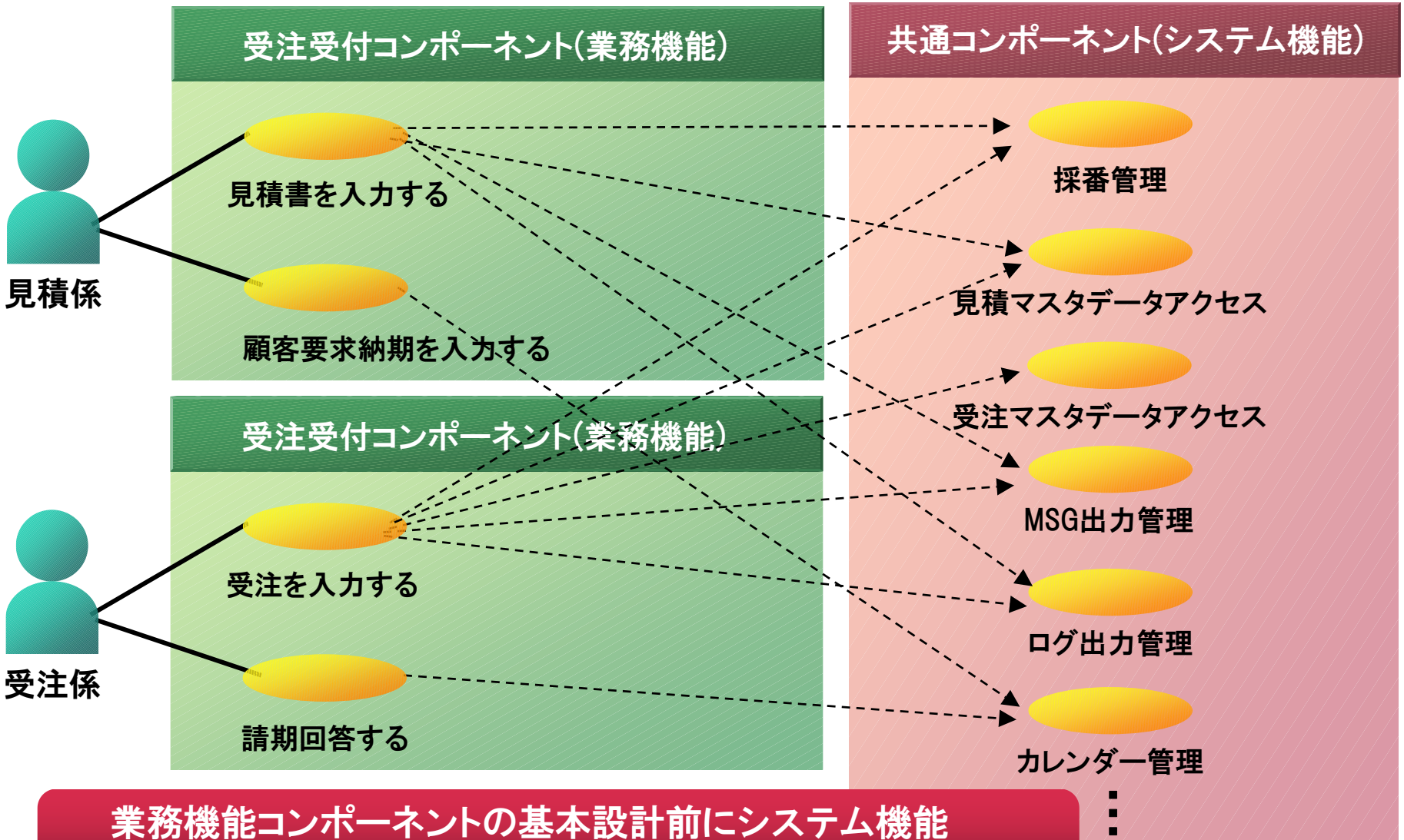
HITACHI
Inspire the Next

# 4-1 システム開発における共通処理機能の重複



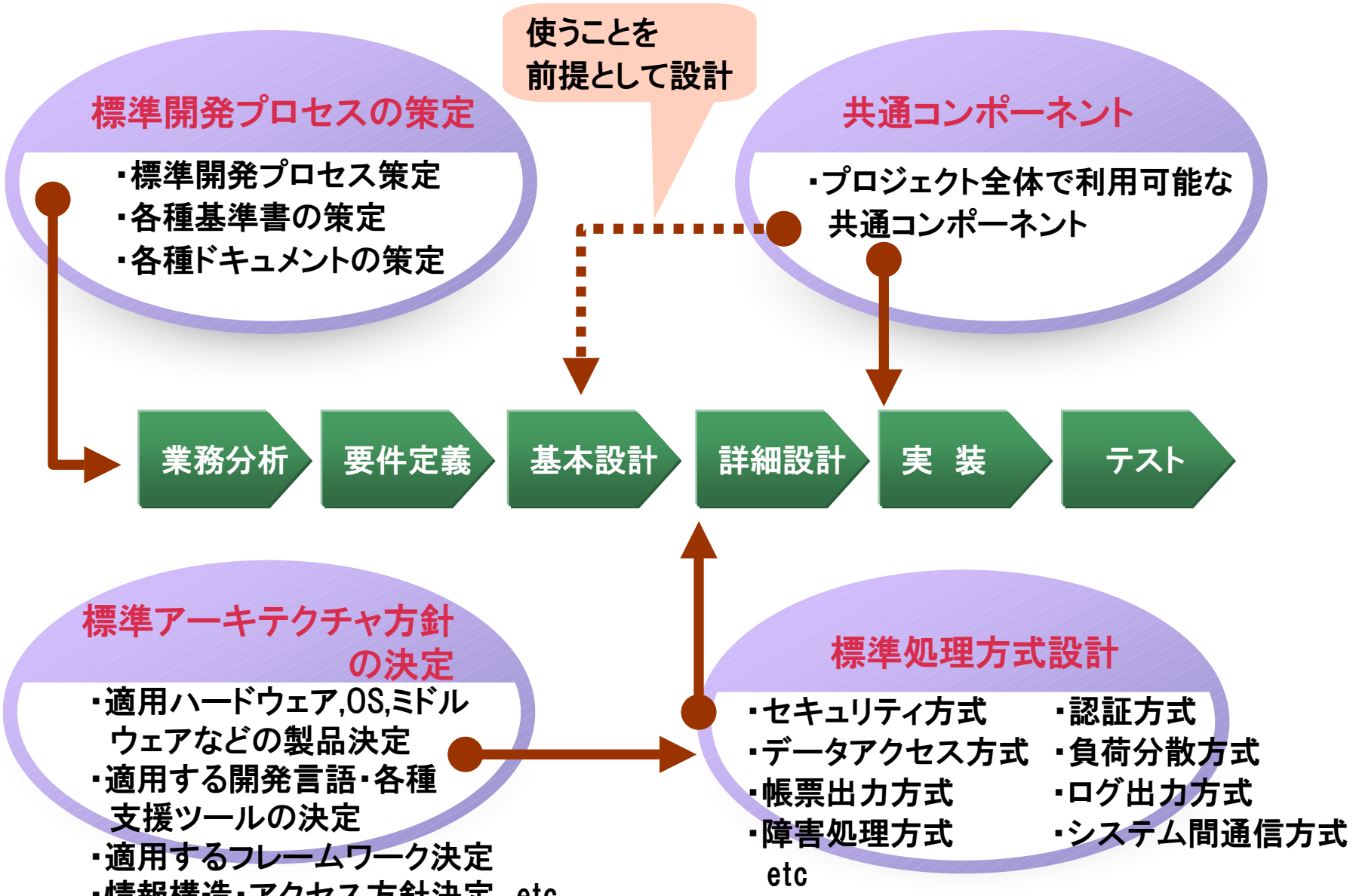

**業務機能コンポーネント基本設計時、システム機能コンポーネント(共通)が決まっていないと、重複して作りこんでしまう危険性が大きい**

# 4-2 共通コンポーネントによる重複開発の回避

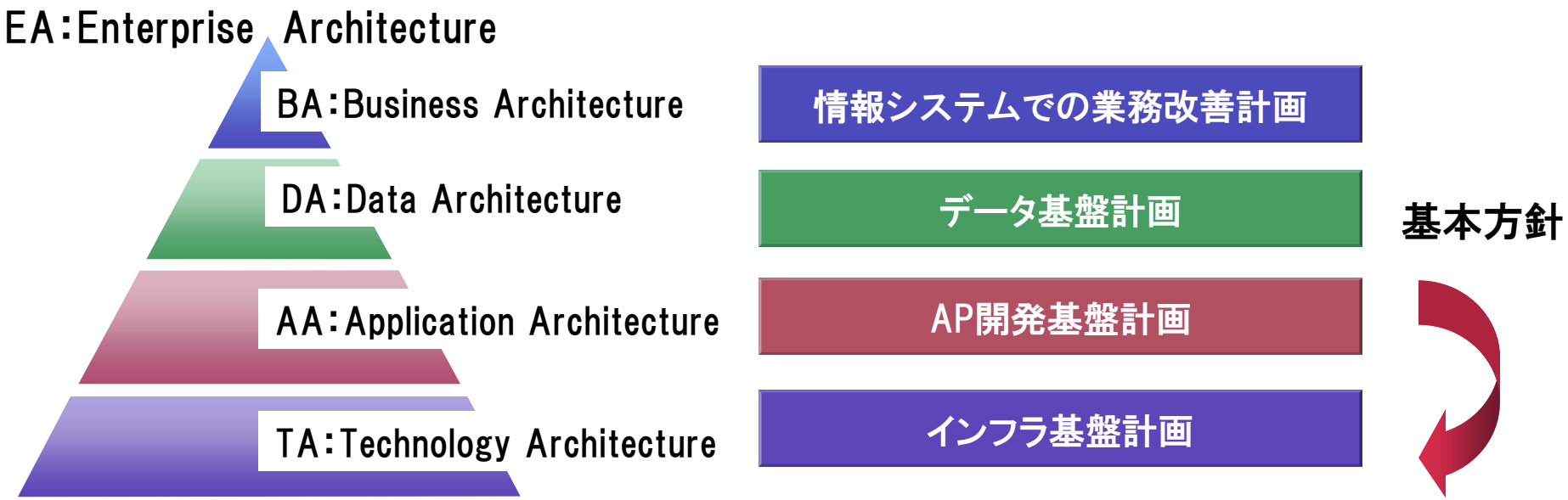



**業務機能コンポーネントの基本設計前にシステム機能コンポーネント(共通)を設計しておき、それを元に設計する**

# 4-3 各種標準化・共通化の決定と工程での位置付け



# 4-4 プロジェクト推進体制の事例

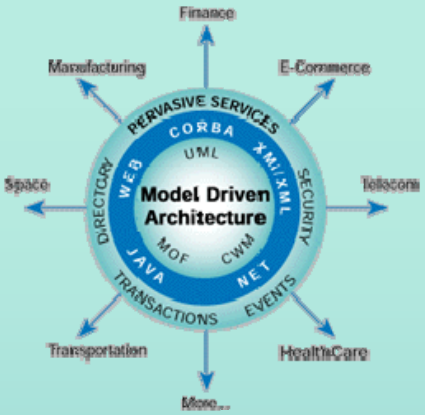



| チーム体制 | 役　　割 |
|---|---|
| 標準化チーム | 標準開発プロセスや各種基準、ドキュメントの標準化、各種アーキテクチャ標準や共通処理方式方針などを決定する |
| 共通コンポーネントチーム | 特にシステム機能コンポーネントの提供と保守、管理、APチームへのデザインレビューへの参画 |
| 共通マスタチーム | 共通マスタの設計とデータアクセスコンポーネントの提供 |
| 運用管理チーム | インフラ環境の構築と各種APの運用管理方式決定 |
| AP開発チーム(複数) | 各種APの開発 |

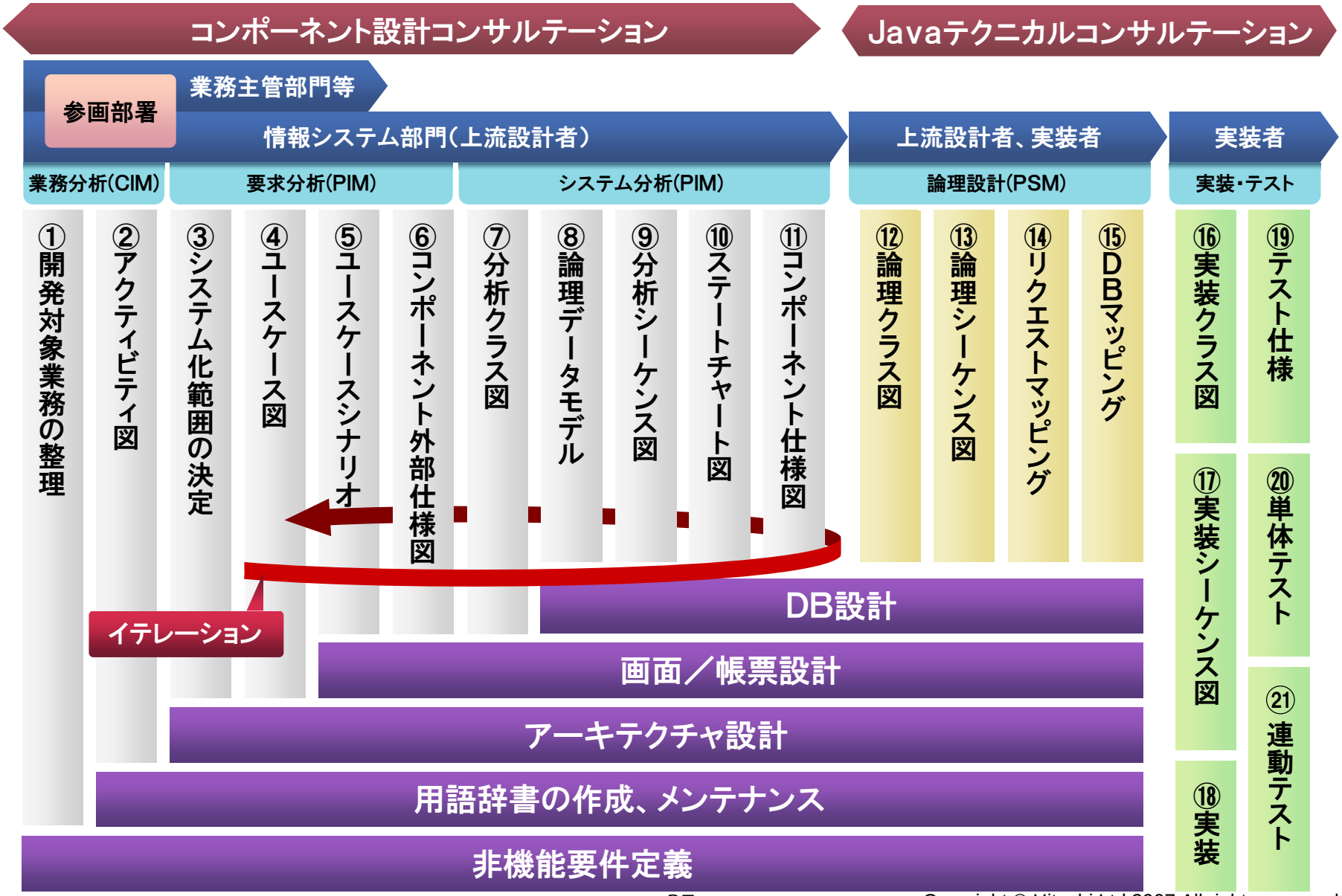
# まとめ

## コンポーネント化においては

- 業務分析からのモデリングが必要
- そのための一貫した分析・開発プロセスと、開発支援などの統一が必要
- MDAベースではモデルそのものの再利用化も可能となる
- 作成したコンポーネントを効率よく管理・運用するための体制と仕掛けが必須




# 参考 コンポーネントベース開発での支援サービス



# 他社権利表示

•OMG, UML, Unified Modeling Language, MDA, Model Driven Architectureは,
Object Management Group, Inc.の米国及びその他の国における登録商標または商標です。
•Borlandのブランド名および製品名はすべて, 米国Borland Software Corporationの
米国およびその 他の国における商標または登録商標です。
•Java 及びすべてのJava関連の商標及びロゴは, 米国及びその他の国における
米国Sun Microsystems, Inc.の商標または登録商標です。
・Oracleは, 米国Oracle Corporationの商標です。

•その他記載の会社名、ブランド名および製品名は、それぞれの会社の商標、もしくは登録商標です。