情報処理学会・経営情報学会
連続セミナー第３回

アプリケーション・アーキテクチャ

２００６年９月５日

NPO法人技術データ管理支援協会
手島 歩三

1. ビジネス・アプリケーションの課題
   - 並列処理
   - データの品質保証
   - ソフトウェア開発・保守アプローチの改革
2. 情報システム・アーキテクチャのビュー：アプリケーション体系
   - アプリケーション・ポートフォリオ（アプリケーション分類）
   - アプリケーション分割
3. アプリケーションのモジュール化
   - モジュール化の必要性
   - データ管理共通基盤
4. ビジネス・アプリケーションの導出
   - 基幹系、情報系
   - パッケージ選択とカスタマイズ
5. 情報基盤整備
6. ビジネス改革と情報システム構築の同期

1. ビジネス・アプリケーションの課題

- 並列処理
  - ビジネスの実世界では多数のステークホールダたちが一見するとランダムに活動している。
  - その活動を「組織の視点」で支援する。
- 組織＝「協働の体系」（C. バーナード）
  - 組織活動の形態：自律・協調・分散
    - 必ずしも命令によって動くだけではない。
    - 異質な能力を持つ専門家たちがそれぞれの持ち場で自発的に行動し、協力し、仕事を成し遂げる。
  - 組織活動の規則性
    - 働き掛ける対象が持つ規則
    - 専門家が果す役割と働き方の規則

情報システムのパラダイム・シフト

- 情報システムの本格的な統合と分散
  - 自律・協調・分散を支える
  - 「もの」の分散、「活動」の現場の分散
- アプリケーションの中心は個々の利用者
  - 利用者が自由にアプリケーションを組み立て、呼び出して利用する。
- 利用者を情報によって繋ぐデータベースと通信
  - 時間と空間の壁を超えて情報を共有し、意思疎通する利用者たち
- ビジネス活動を支援する個別アプリケーション
  - ビジネス活動の事実を採取し、データベースを更新する。
  - 蓄積されたデータベースを参照し、課題の状況を把握し、遅れ同期（asynchronous）する専門家たち
  - 必ずしも手続的に連続ていない。

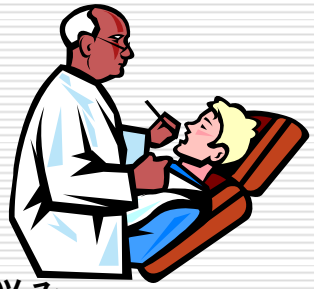
アプリケーション・システムの現状

- 業務用ソフトウェアの諸問題
  - アプリケーション間の食い違い
    - 機能重複とデータ仕様の相違
  - 同じ機能に関するアプリケーション・モジュールが複数存在し、変更・改修の漏れが生じる。
  - データ品質保証の根拠と方法が異なる。
- 業務用ソフトウェア開発の諸問題
  - ユーザ要求の食い違いと頻繁な変更
  - ユーザ要求の裏に隠れている共通事項
  - 不明確なテスト基準：機能だけでよいのか？

- 業務用ソフトウェア設計・導入の正当な根拠はあるのか？

アプリケーション・システム構築の困難

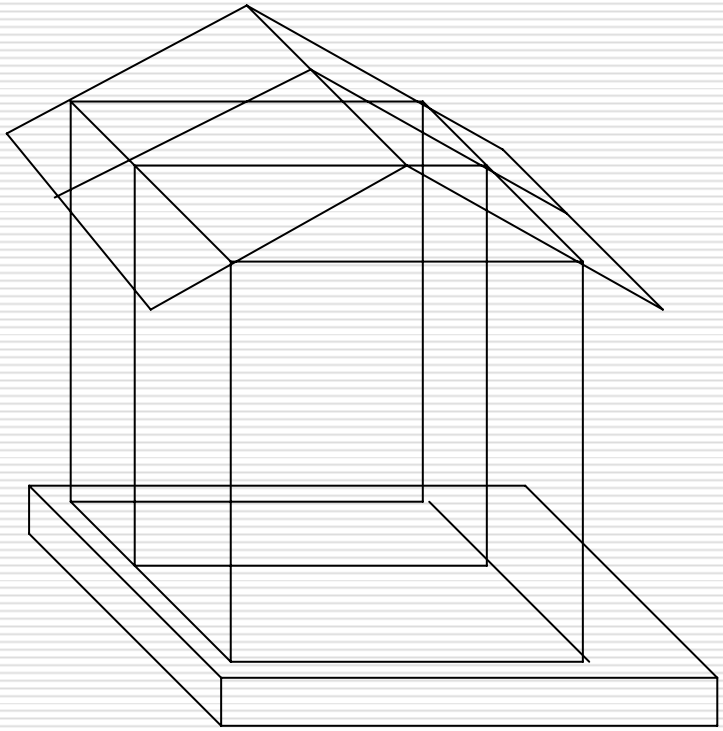
- 開発者の悩み
  - 同じようなソフトウェアを繰り返し開発することに飽きた
  - ユーザ要求に含まれていない重要なことに気付いているが、言うと作業量が増え、しかも費用をもらえない
  - 他人の仕事なので、口を挟めないが、同僚が他のアプリケーションの中で同じ機能を異なる考え方で実装している。
  - 重要なアプリケーションを開発すると、何かトラブルが起きるたびに呼び出され、責任の所在を明らかにするために他人の担当部分の問題を調べさせられる



- ユーザ企業の悩み
  - 要求分析・要求定義段階で漏れなく、正確に要求を述べることは極めて難しい
  - 本稼働開始までに組織の内外で変化が起き、要求変更を避けられない
  - 変更内容によってソフトウェア開発の納期延長と費用増加の程度を推測したいが、判断基準がない

情報システム・アーキテクチャ

- 情報システムには骨格がある。
  - 何を以て骨とし、何を以て肉とするか、正確に認識する必要がある。
  - 情報システムの骨格は「情報」体系にほかならない。
  - 情報を採取し、蓄積し、参照するために情報処理機能を用意する。この部分が筋肉に相当する。
  - 情報の利用者が使いやすいように情報を加工し、表現する部分が皮膚に相当する。
- 実現手段（材料と）工法
  - 情報システムの構成要素によって利用する材料や工法が異なる。



何に基づいて情報体系を設計するか、正当な論拠が必須！

2. 情報システム・アーキテクチャのビュー
アプリケーション体系

- 情報技術適用業務体系と業務用ソフトウェア構造
- 組織構造（ビジネス・アーキテクチャ）との適切な対応
  - 共通事項の統合管理
  - 個別事項の個別管理
  - 組織の分権と統合に対応するデータとアプリケーションの集中・分散
- アプリケーションの特性に合う情報技術の選択
  - 市販の汎用アプリケーション・パッケージの応用
  - 業務用パッケージの適切な選択
  - 自社開発すべきアプリケーションの特定
  - 情報基盤技術の適切な選択
    - 稼働環境の選択による開発範囲の限定と処理効率向上および品質保証
    - 開発環境の選択による開発の品質保証とソフトウェア構造の整合

アプリケーション・アーキテクチャ設計時の配慮

- データの共同利用
- 組織の業務形態と基盤技術整合
- 基盤技術商品のバージョンアップ対策
- 同一機能に関するモジュールの再利用
- 類似機能に関するアルゴリズムの再利用
- データの品質保証
- 新アプリケーションへの移行可能性保証
- テスト可能性保証
- ビジネス改革活動とアプリケーション構築の同期
- 開発すべきことと、再利用すべきことの明示

アプリケーション・ポートフォリオ
情報処理形態によるアプリケーション分類

基幹系アプリケーション

- Mission Critical Application
  - ビジネス活動を支援し、その実をトランザクションデータとして採取し、データベースを更新する
  - 止まると営業中止と見なされる
- Logistics Application
  - 蓄積されたデータを分析し、最適化を図る

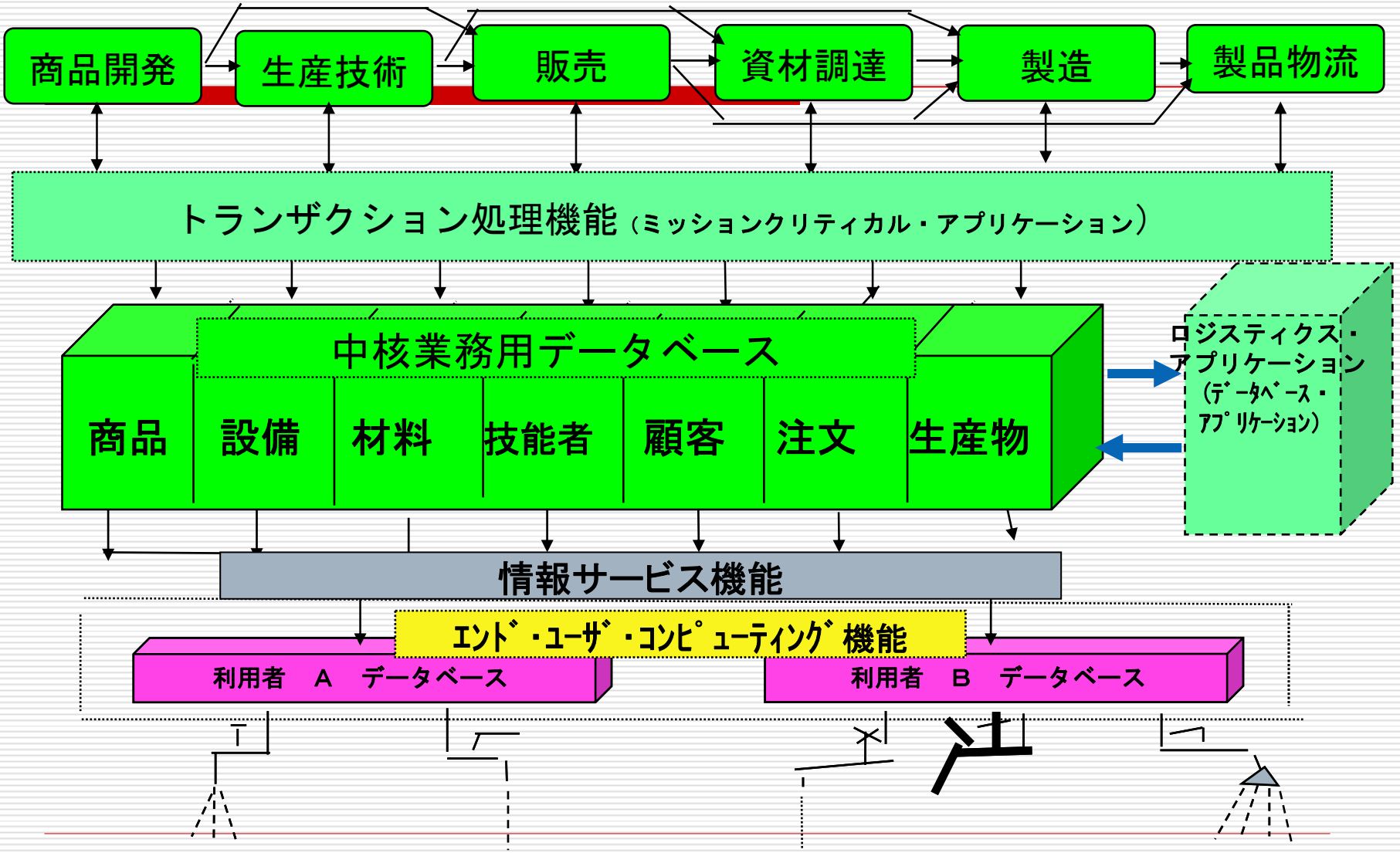
情報系アプリケーション

- Information Service
  - 基幹系データベースおよびトランザクションデータのログからデータを抽出し、利用者データベースに渡す。逆流は許さない。
- End User Computing
  - 利用者が自らの手でデータを加工し、欲しい情報を取り出す。

その他のアプリケーション

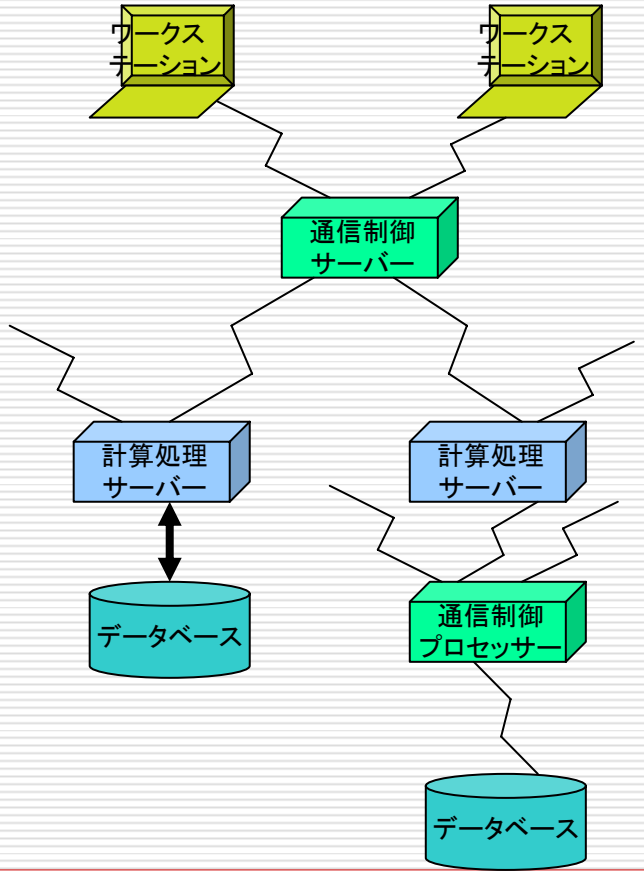
- Office Service Applications：電子メール、電子会議、個人データ処理など
- Front end Applications：利用者に接し、基幹アプリケーション利用を助ける。
- 対外接続系アプリケーション：異なる組織の基幹アプリケーションをデータで繋ぐ。

情報システムの概念的構造



分散処理と幾つかの処理形態

- 利用者の所在地の分散
- データベースの分散
- 処理要求と処理形態
  - バッチ処理
    - リモートバッチ処理を含む
  - オンライン・トランザクション処理（OLTP)
    - 同期方式 (synchronous)
    - 非同期方式 (asynchronous)
  - 会話型処理
  - ファイル転送
  - プログラム間通信（PTP）



岩田裕道氏教育
資料より引用

7つの基本モデル

	データ / 相互作用	集中	分散	
			垂直	水平
同期	トランザクション型	TRX1	TRX2	
同期	依頼応答型	IA1	IA2	
非同期	遅延型	AS1	AS2	AS3

- バッチ処理はクライアント／サーバ処理形態を採らないため除いている
- 上記のような「プル型」の処理に加え，最近は「プッシュ型」の処理も注目されている.

岩田裕道氏教育
資料より引用

7つの基本モデル

①TRX1　集中トランザクション処理
②TRX2　分散トランザクション処理
③IA1　遠隔データアクセス
④IA2　分散データアクセス
⑤AS1　ローカル・メッセージング
⑥AS2　データ・ステージング
⑦AS3　グローバル・メッセージング

岩田裕道氏教育
資料より引用

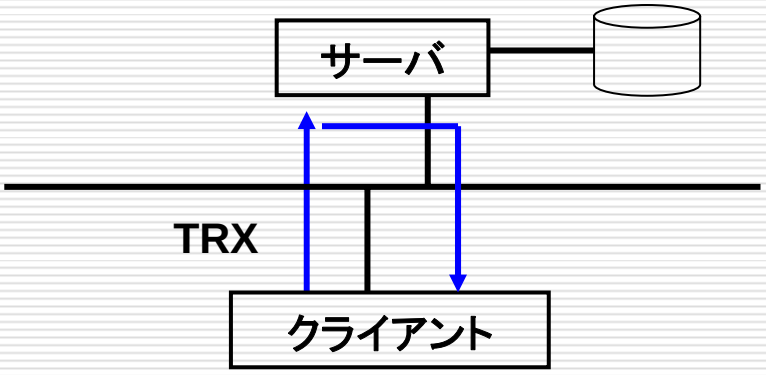
6．3　トランザクション型

①集中トランザクション処理（TRX1）

- 単一データベースでの追加・更新主体のトランザクション処理に適する
- 一般にTPモニターなどのミドルウエアを使って実装する
- 負荷が大きい場合は，サーバがアプリケーション・サーバとデータベース・サーバに分割される

アプリケーション例：

受発注管理，在庫管理，株式注文処理，生産管理，小売りPOS，銀行の勘定系，座席予約



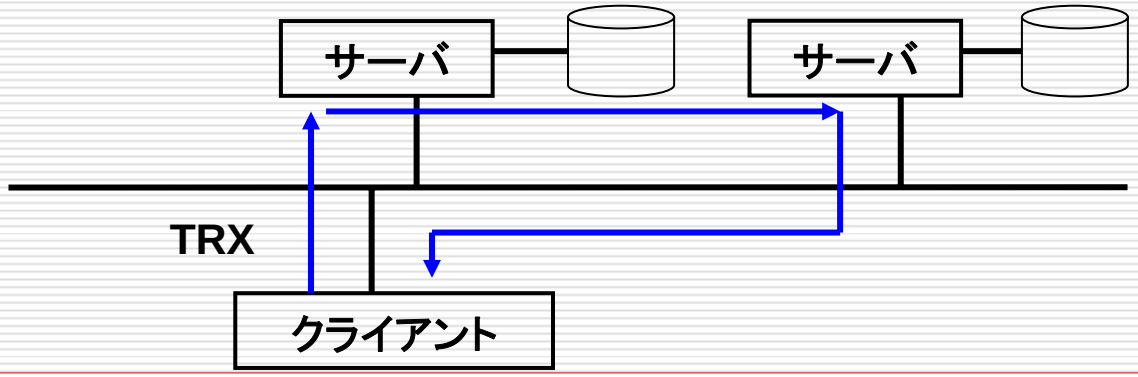
岩田裕道氏教育
資料より引用

②分散トランザクション処理（TRX2）

- １つのトランザクションの中で，複数のデータベースへの更新や問い合わせを同時に実行する複雑なトランザクション処理
- ２相コミットなどの技術を必要とし，実装は①に比べて高度で複雑になる
- 負荷が大きい場合は，サーバがアプリケーション・サーバとデータベース・サーバに分割される

アプリケーション例：

銀行の勘定系，旅行の手配・予約，仕入れ・在庫と直結した受発注管理



岩田裕道氏教育
資料より引用

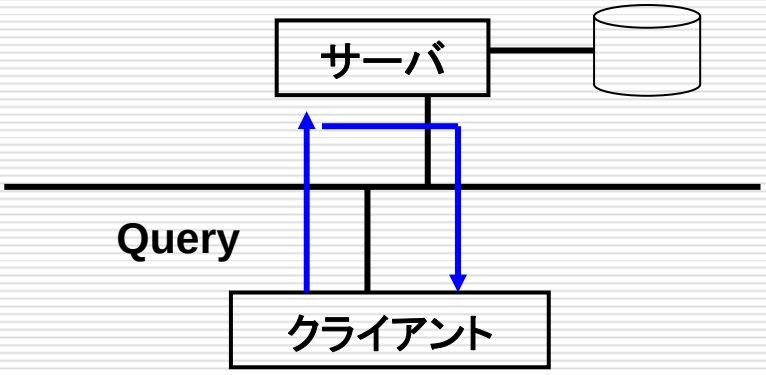
6．4　依頼応答型

③遠隔データアクセス（IA1）

- 単一のデータベースに対する，SQLなどによる問い合わせ中心の即時処理に適する
- 負荷の大きいクライアント・アプリケーションや，複数のクライアントに共通のアプリケーションなどはサーバに置かれる

アプリケーション例：

営業支援，各種照会処理，情報提供サービス．その他に情報系アプリケーション（情報サービスとEUC）の多くはこのタイプの処理となる．



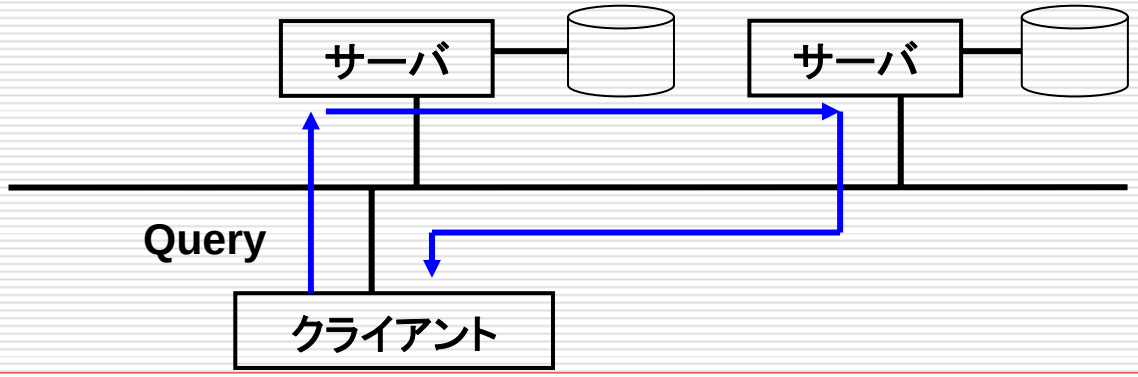
岩田裕道氏教育
資料より引用

④分散データアクセス（IA2）

- 複数のデータベースやファイルを同時にアクセスする必要がある，問い合わせ中心の即時処理に適する
- 複雑な問い合わせの場合は，分散問い合わせ最適化などの機能を備える必要がある

アプリケーション例：

全社売り上げ統計，全社レベルの生産性データ分析，複数箇所にわたる在庫分析



岩田裕道氏教育
資料より引用

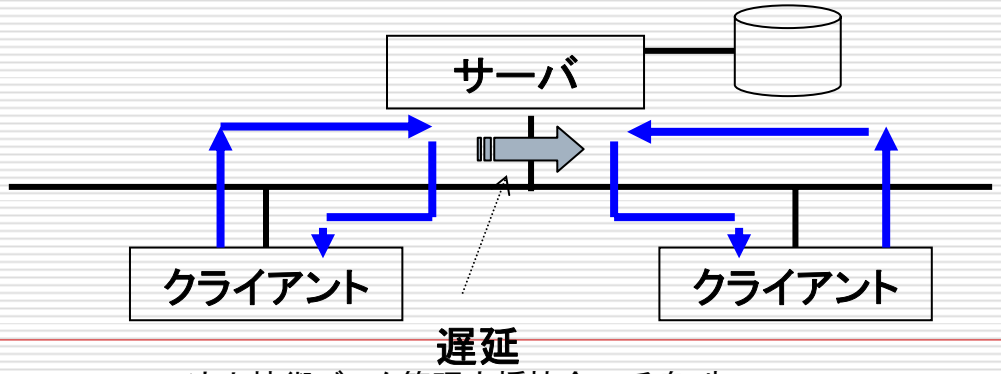
6．5　遅延型

⑤ローカル・メッセージング（AS1）

- メール／文書／写真／伝票など多様な形式の情報を，部門サーバにあるグループウエア／文書管理ツール／ワークフロー・ツールなどを使用して多数のクライアントが共有し交換する
- サーバの処理は一般に，クライアントの要求に非同期で（遅延して）実行される

アプリケーション例：

電子メール，伝票／報告書などの配送と回付，イベントの通知，文書の保管，業務フローの実行／監視／報告



岩田裕道氏教育
資料より引用

⑥データ・ステージング（AS2）

- 階層的に分散したデータの，非同期な共有による情報の管理と有効活用を促進するアプリケーションに適する.
- 典型的には，企業サーバと部門サーバがデータを通じて疎結合し，必要に応じてアップロードやダウンロードしてアプリケーションを実行するような場合

アプリケーション例：

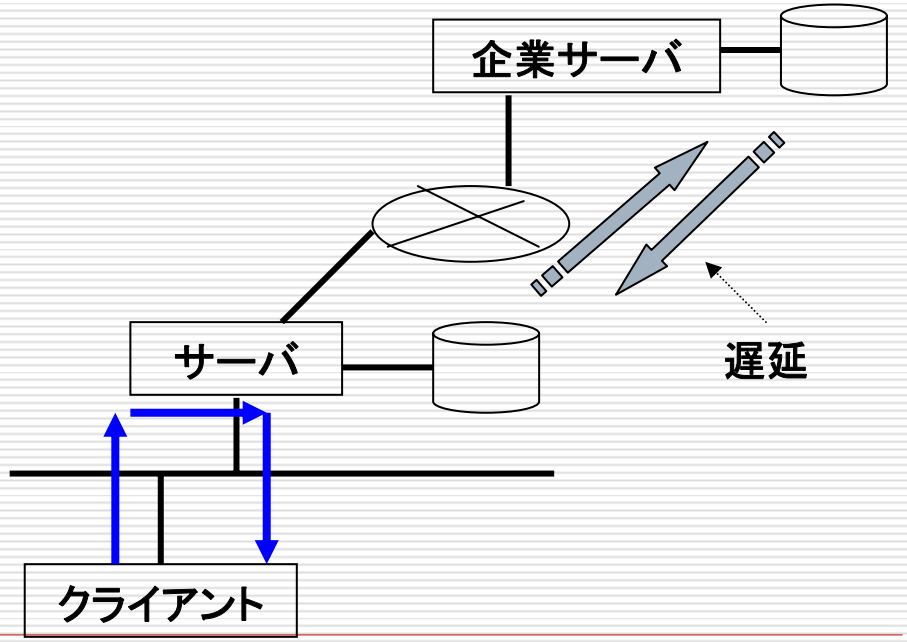
受注データのアップロード，

マスターの一部のダウンロード，

大福帳やデータウエアハウス，

本支店にまたがる人事・経理

システム



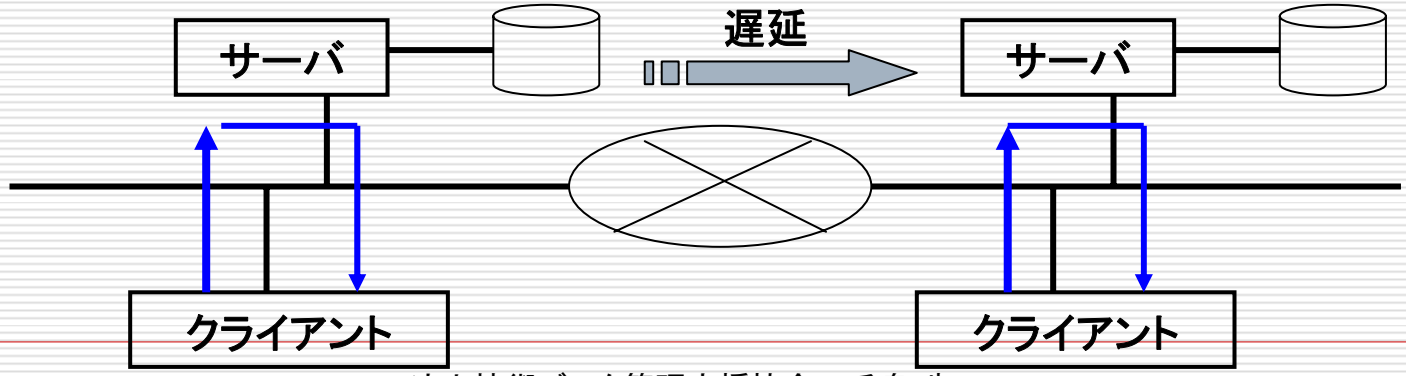
岩田裕道氏教育
資料より引用

⑦グローバル・メッセージング（AS3）

- 企業内で複数の部門サーバが，データやメッセージを通じて疎結合し，必要に応じてアプリケーションの連携を行うのに適する
- 異なる複数の企業間で，データやメッセージを通じて情報システムの疎な統合を行うのに適する
- WEBによるインタネットやエクストラネットは，一般にこの形態を採る

アプリケーション例：

全社レベルの電子メールやワークフロー，受注・在庫・仕入れ・出荷・請求の一貫処理，生産計画・工程管理・在庫管理の統合，旅行の手配・予約



岩田裕道氏教育
資料より引用

参考： ポートフォリオとの対応

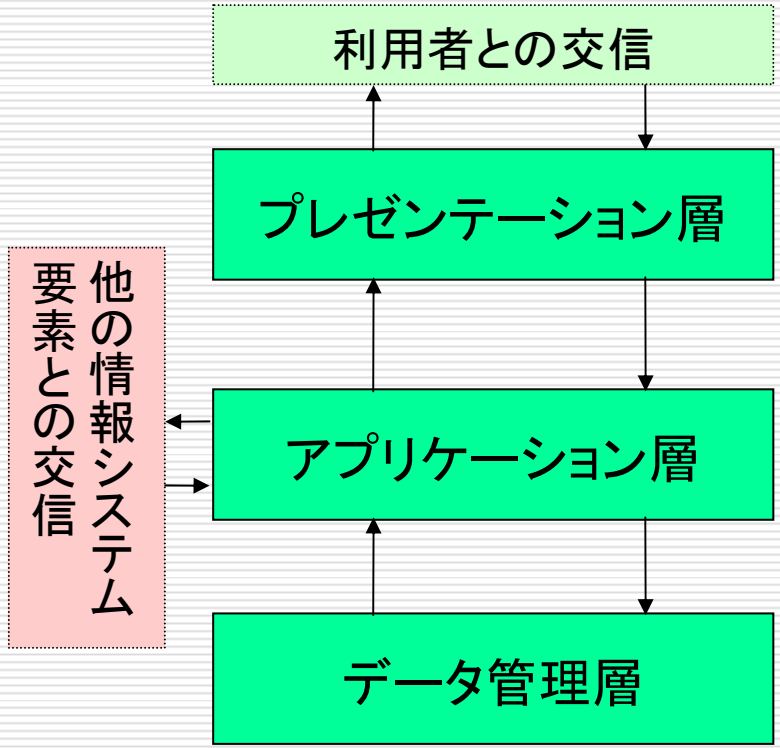
アプリケーション・ポートフォリオの各系に属するアプリケーションは，どの基本モデルに対応するか？

系	対応する基本モデル
基幹系	・集中トランザクション処理（TRX1） ・分散トランザクション処理（TRX2）
情報系	・遠隔データアクセス（IA1） ・分散データアクセス（IA2） ・データ・ステージング（AS2）
オフィス支援系	・ローカル・メッセージング（AS1） ・グローバル・メッセージグ（AS3）
連合支援系	・データ・ステージング（AS2） ・グローバル・メッセージング（AS3）

アプリケーション分割
Application Partitioning

- 対話型orトランザクション処理型
  - プレゼンテーション層
    - 「活動」に必要なデータの表示とインプット
    - 操作ガイド
  - アプリケーション層
    - データ変換
  - データ管理層
- バッチ処理
  - モジュール
  - プログラムの骨格
  - データ管理

アプリケーション分割
application partitioning

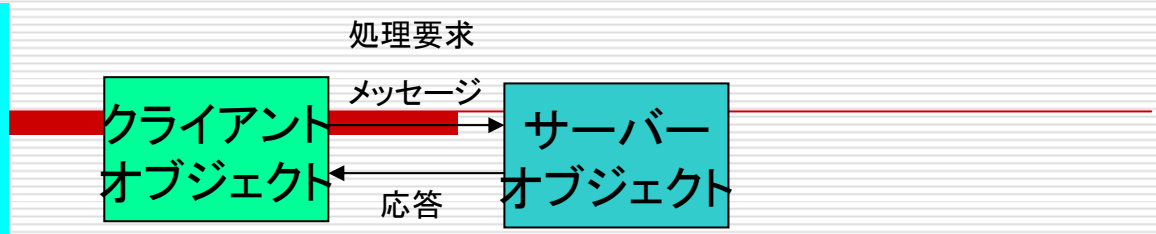


- 要求変更がプログラムに及す影響を最小限に食い止めるための方策
- ユーザ要求の性質に応じて、一台のコンピュータの中でのアプリケーション要素を分割
- 層毎にコンピュータを割り当てると通信量が激増し、処理効率と品質保証可能性が低下する恐れがある

情報システムの構造変化

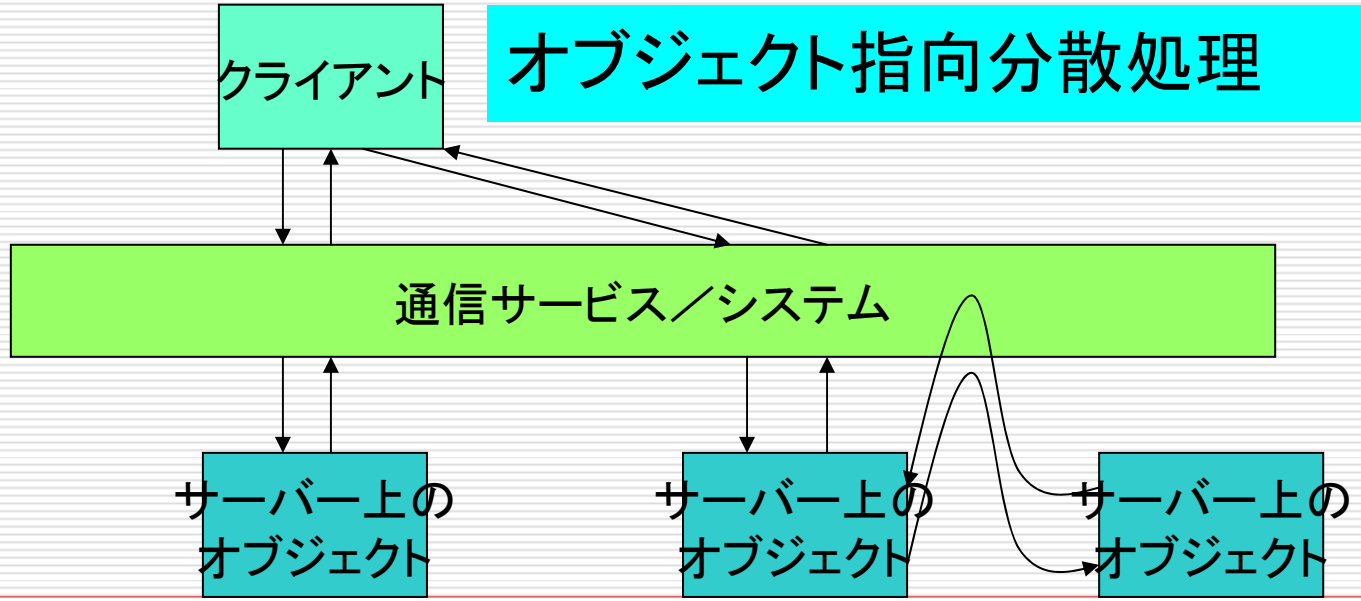
- 独立性の高いアプリケーション・プログラム
  - 実体データ管理オブジェクト群
  - 活動支援エージェント群
- 希薄になったアプリケーション・システム／サブシステム概念
  - 従来もシステム／サブシステムの範囲はプロジェクトの都合によって決められてきた
  - 実際には共通部分が存在し、境界は明確でない
- アプリケーションの中心の移動
  - ユーザ個人によるアプリケーション組み立て
  - 共通モジュールを利用することによる自然な規則遵守

オブジェクト指向
クライアント／サーバ概念

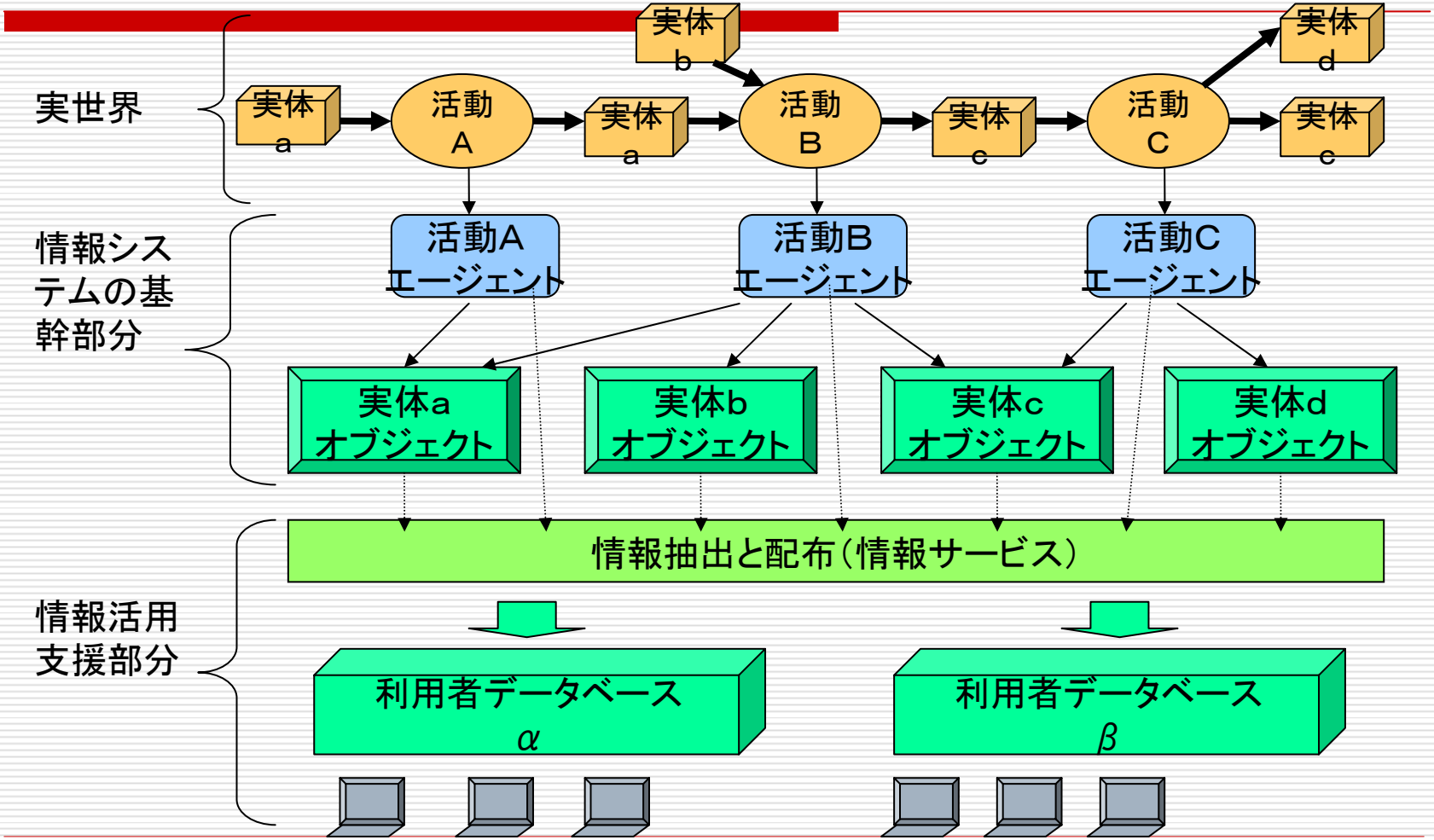


（実質的には
エージェント）

オブジェクト指向分散処理



オブジェクト指向情報システムの概念的構造



３．アプリケーションのモジュール化
３．１　モジュール化の必要性

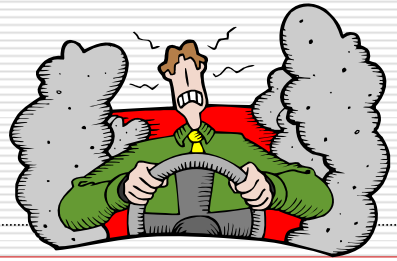
- 保守性に関する配慮
  - **開発費用に比べて、保守費用の期間、労力、費用が増加する**
  - 開発作業の大半はテストと修正であり、保守作業と共通点が多い
  - **適切なモジュール化と再利用を考慮する必要がある**
  - **局所化**

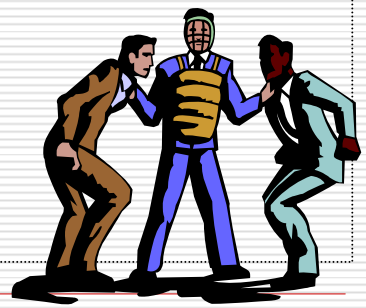
- 大皿に山盛りのスパゲティ

ル化は情報システム構造を複雑化させる

- **副作用！**

ある部分機能を変更すると他のモジュールに影響が及ぶ



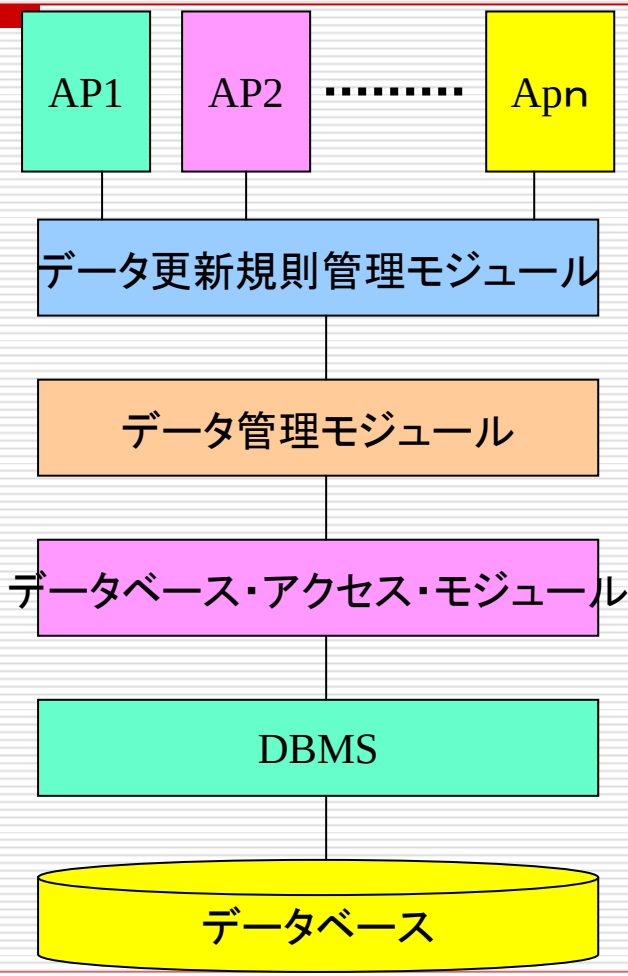


３．２　モジュール化

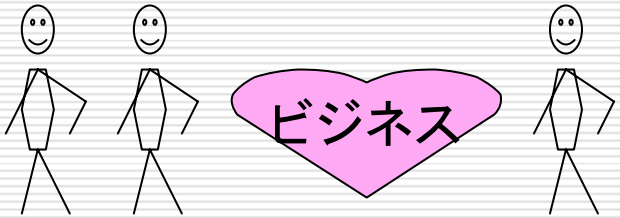
- **共通機能**
  - データ仕様とデータ構造が同じ
  - **モジュール再利用が必須（mandatory）**
- **類似機能**
  - データ仕様が異なり、データ構造が同じ
  - **アルゴリズム再利用が好ましい**
    - データ仕様を外部から与える
      - カスタマイズ機能が必須
    - 外部でデータ仕様を変換して合わせる
      - 副作用：データ変換による処理効率低下

３．３　データ管理共通基盤

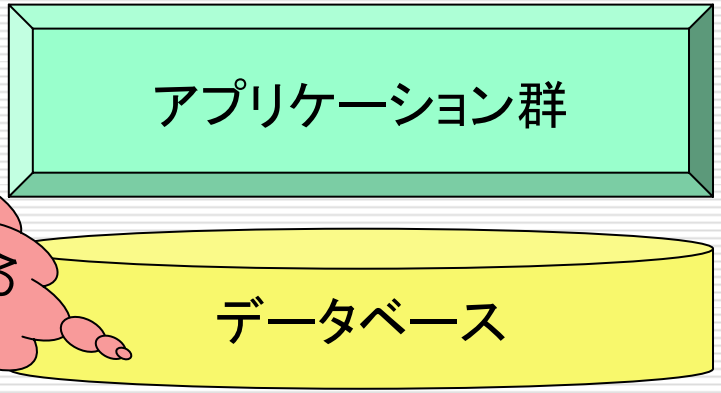
- アプリケーション・インターフェース
- 状態遷移規則によるトランザクション・データの品質チェック
- 参照制約による更新データの品質チェック
- データの検索・参照サービス
- 特定DBMSから独立したデータアクセス
- 特定DBMSによるデータ管理



都市計画アプローチ

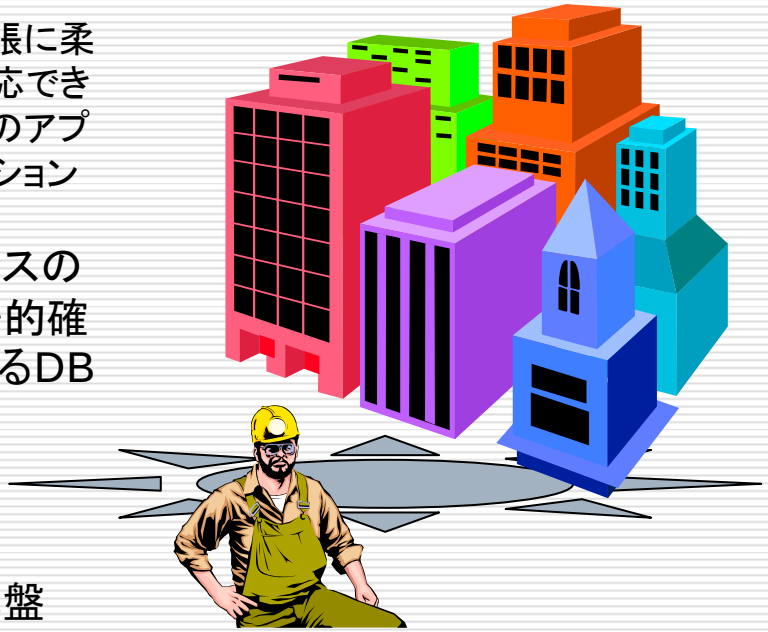


環境に働き掛け進化するビジネス



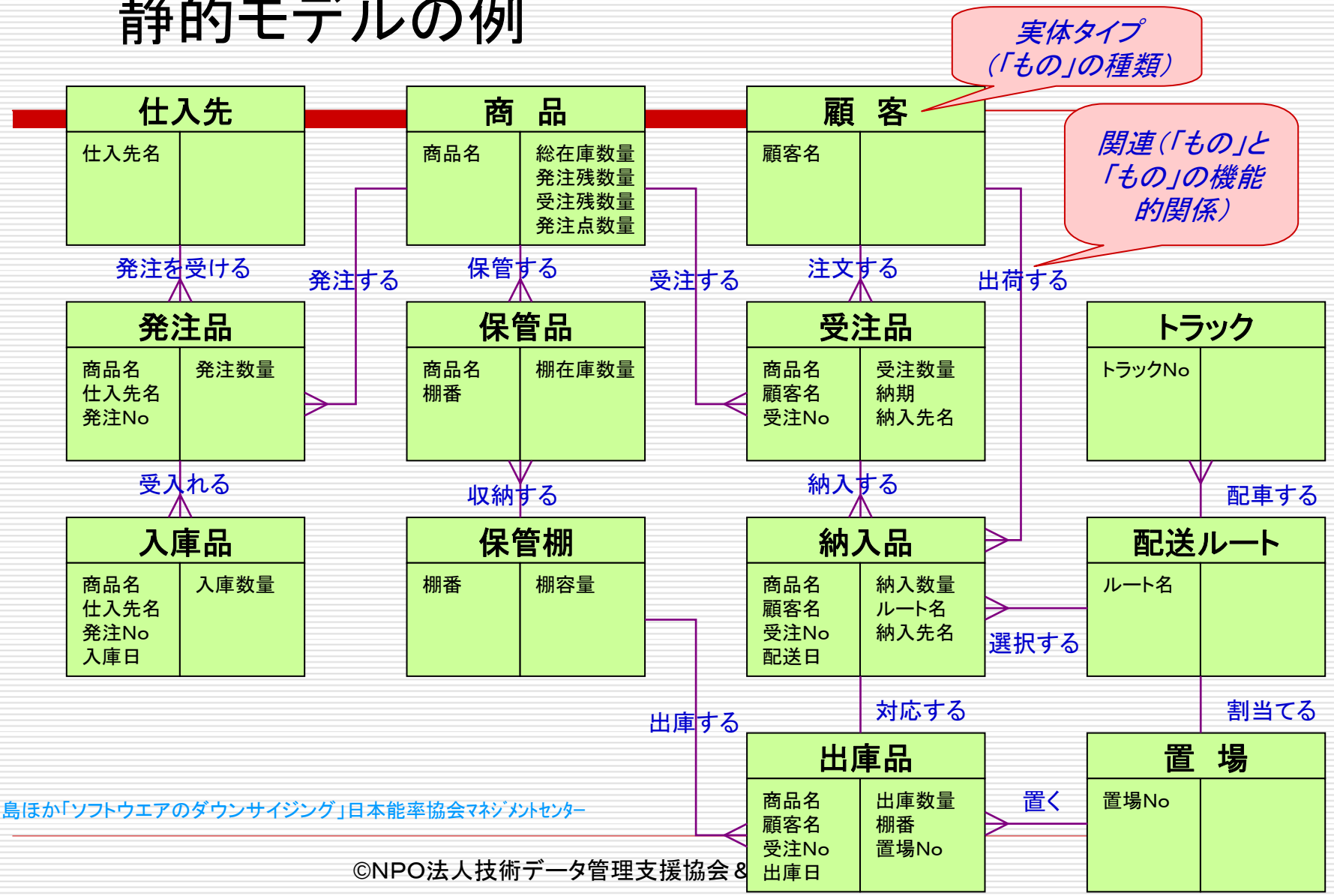
変更拡張に柔軟に対応できる構造のアプリケーション

ビジネスの事実を的確に捉えるDB



急速に発達する情報技術を柔軟に取り組む情報基盤

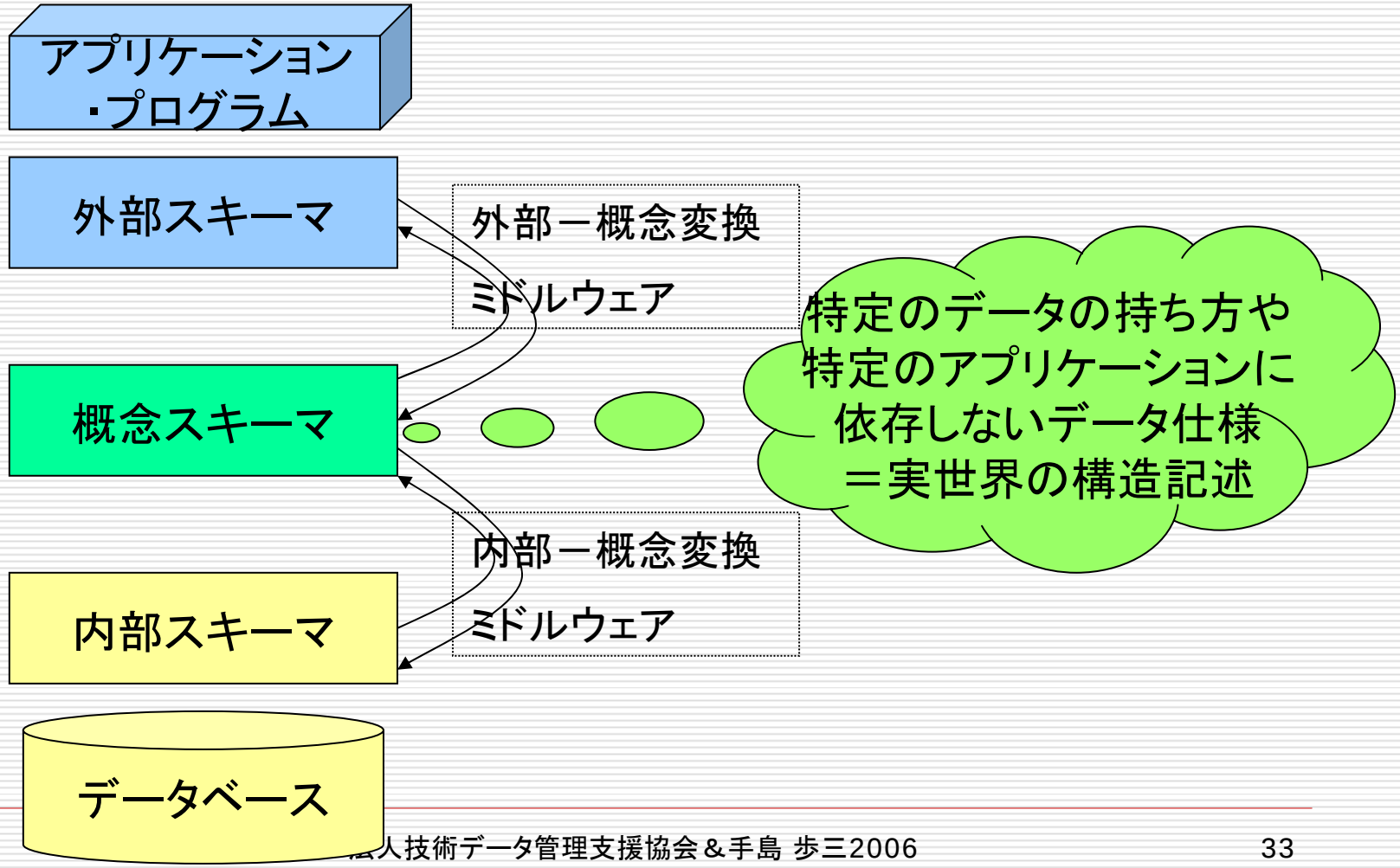
静的モデルの例



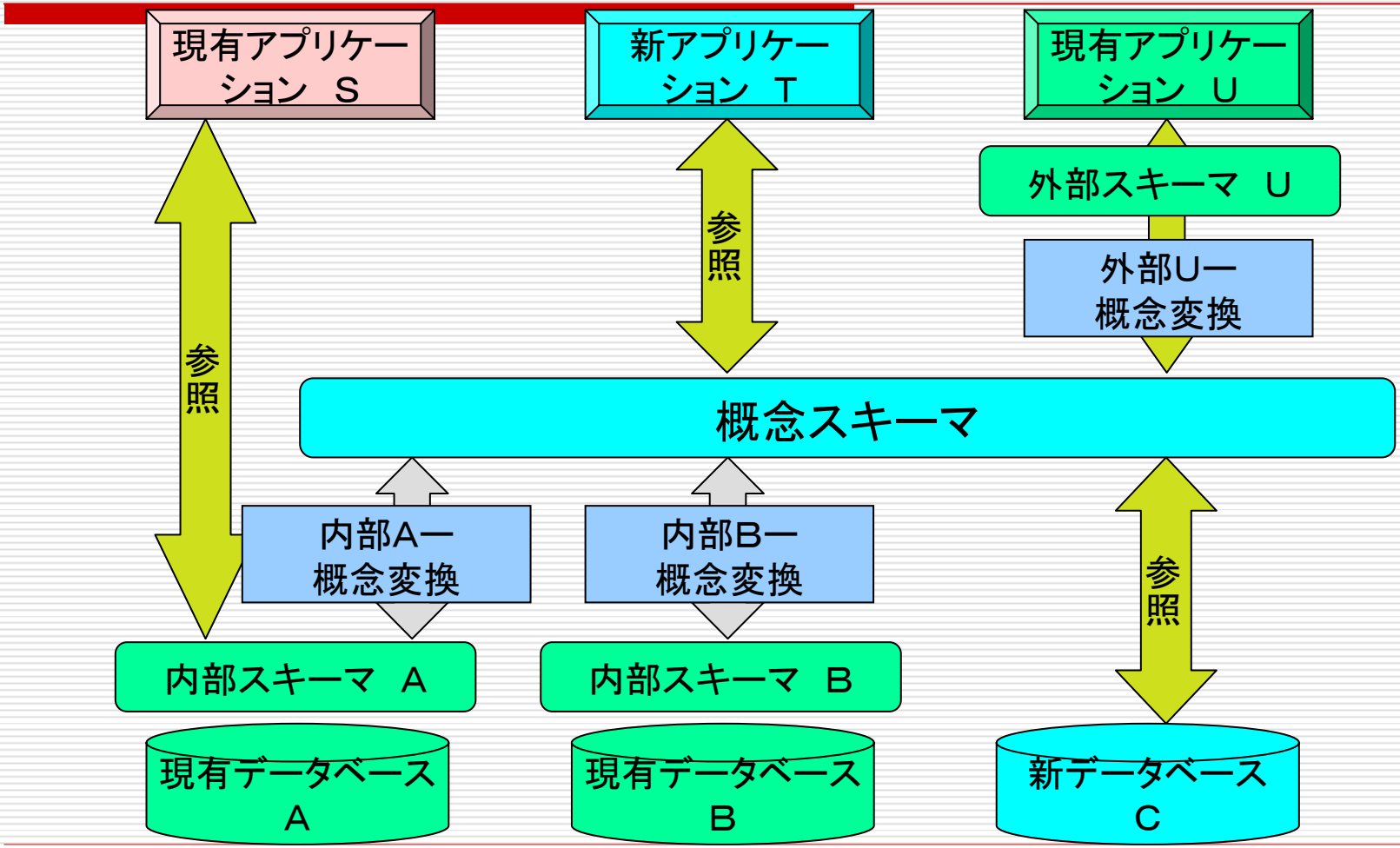
引用：手島ほか「ソフトウエアのダウンサイジング」日本能率協会マネジメントセンター

三層スキーマ概念

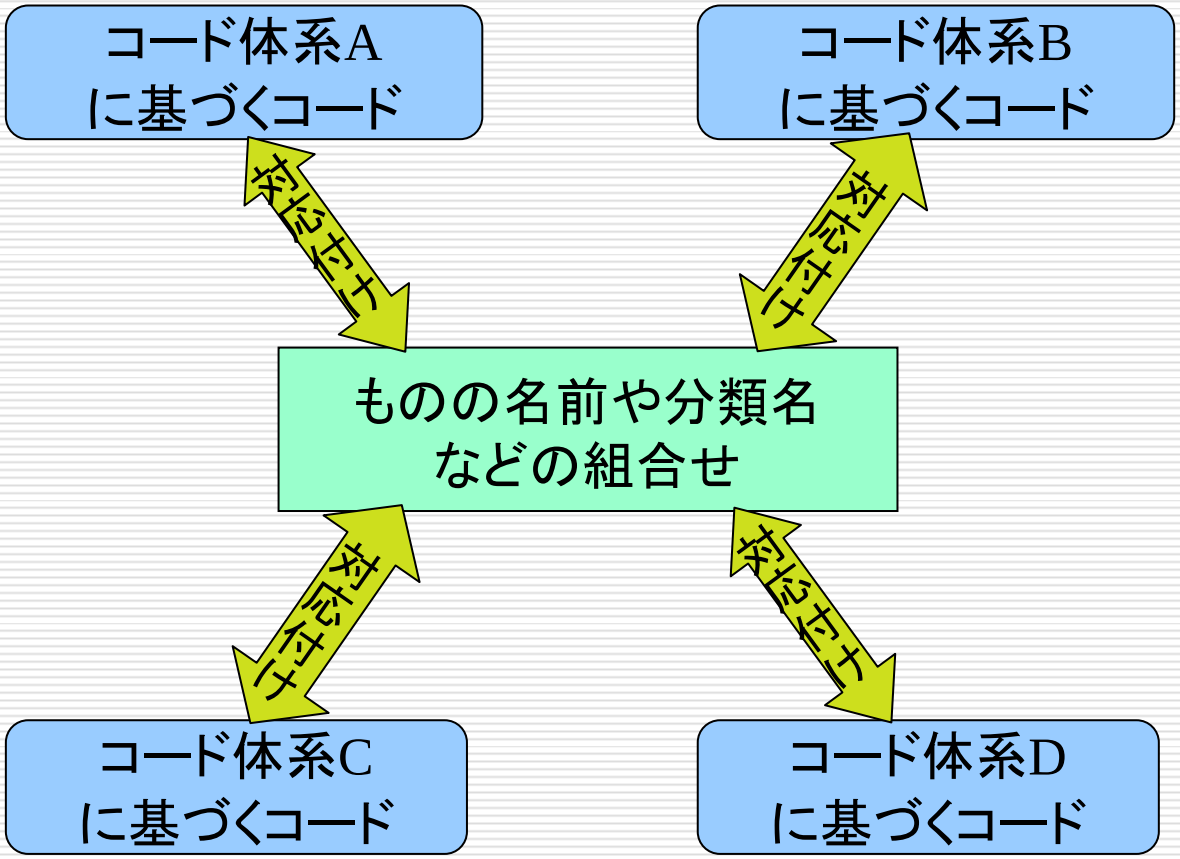
（“The ANSI/APRAC DBMS Model”1976, を参照し解釈を加えた）



3層スキーマ概念によるデータ仕様統合



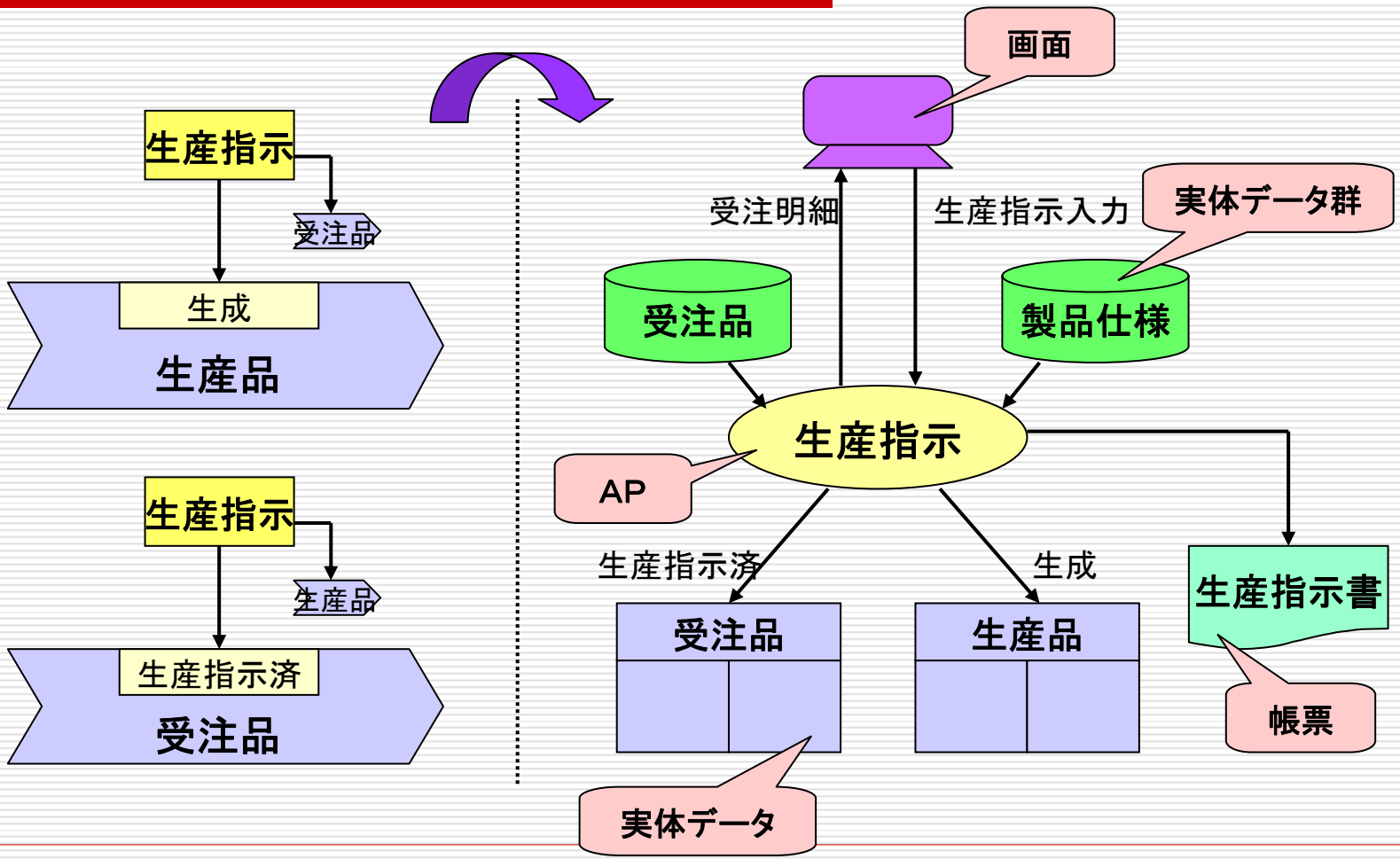
コードレス・コードの仲介によるコード変換（ミドルウェア開発可能）



４．ビジネス・アプリケーションの導出
４．１　基幹系アプリケーション導出

- データ構造の崩れを補うアプリケーション導出
  - 「活動」は「もの」の状態を変化させる
  - 「活動」単位に行われる情報処理
  - 一つの「活動」で複数の「もの」のの状態が変化することがある
  - アプリケーション機能のアウトラインを描く（Context　Diagram）
- アプリケーション・ポートフォリオの当てはめ
  - ビジネス活動が行われるタイミングと情報処理形態
- データ品質保証のための補足
  - 計画／現物データ生成のためのマスターデータ参照
  - インプットデータのチェックのための計画／現物データ参照
  - トレーサビリティ保証のためのアウトプットデータ生成

概念データモデルより
基幹アプリケーション機能を導出する（例）



トランザクション処理の設計
情報対象物と出来事／活動との関係

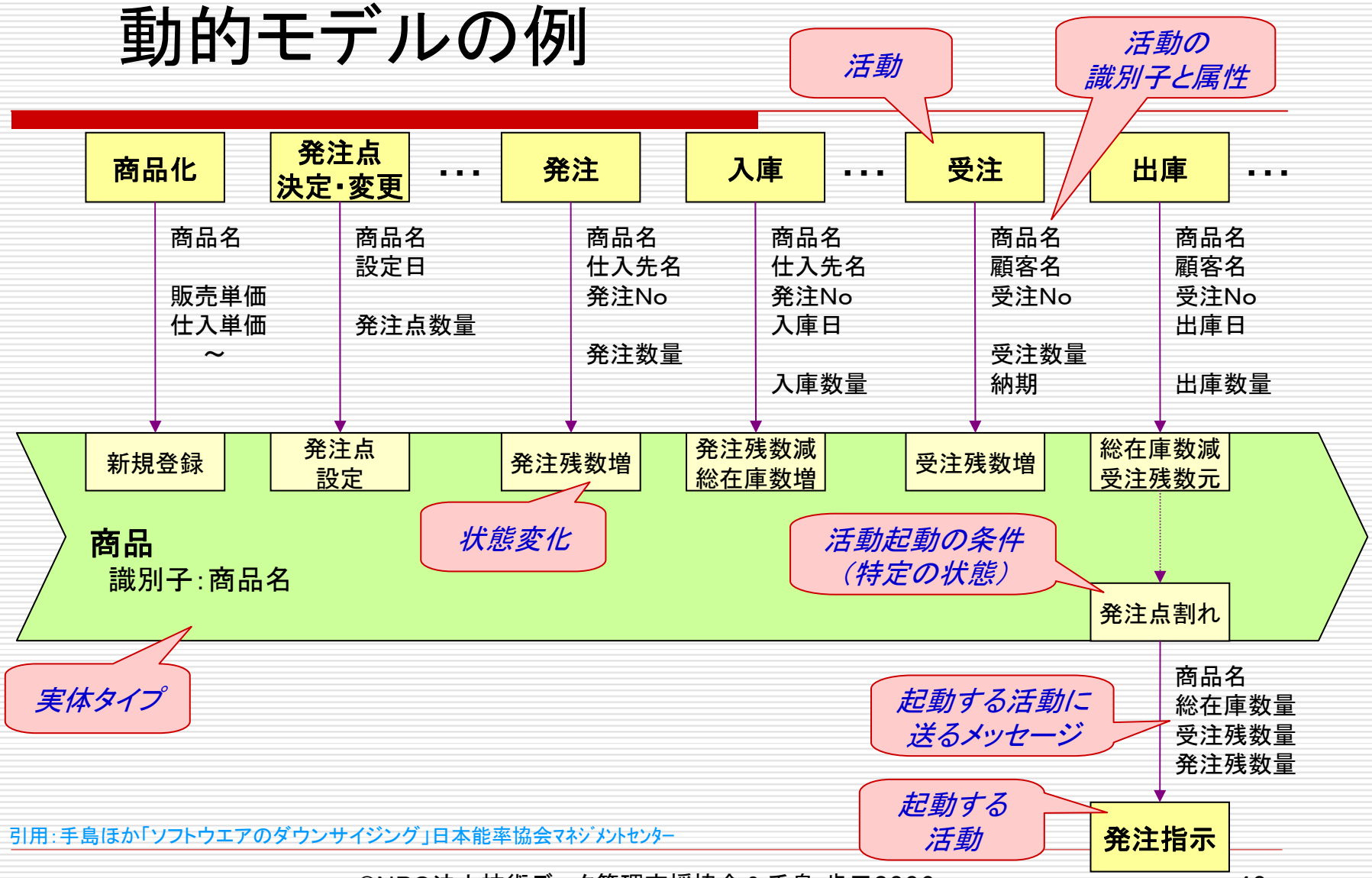
- 対象世界の変化規則
  - 対象世界に個々の物が出現し、変化し、消滅するまでの変化規則を捉える
- 物の状態を変化させる活動とその順序
  - 物識別子
  - 活動識別子（以下複数）
  - 活動属性
  - 活動に伴う物データ更新
- 物の状態によって起動する活動
  - 起動条件

動的性質による「もの」の分類

- クラス：「もの」を形や色などの属性でなく、動的性質の共通性によって分類する。
  - 例：乗用車、部品メーカー
- サブクラス：動的性質の一部分が違っている場合、部分が違うクラスとして細分化する。

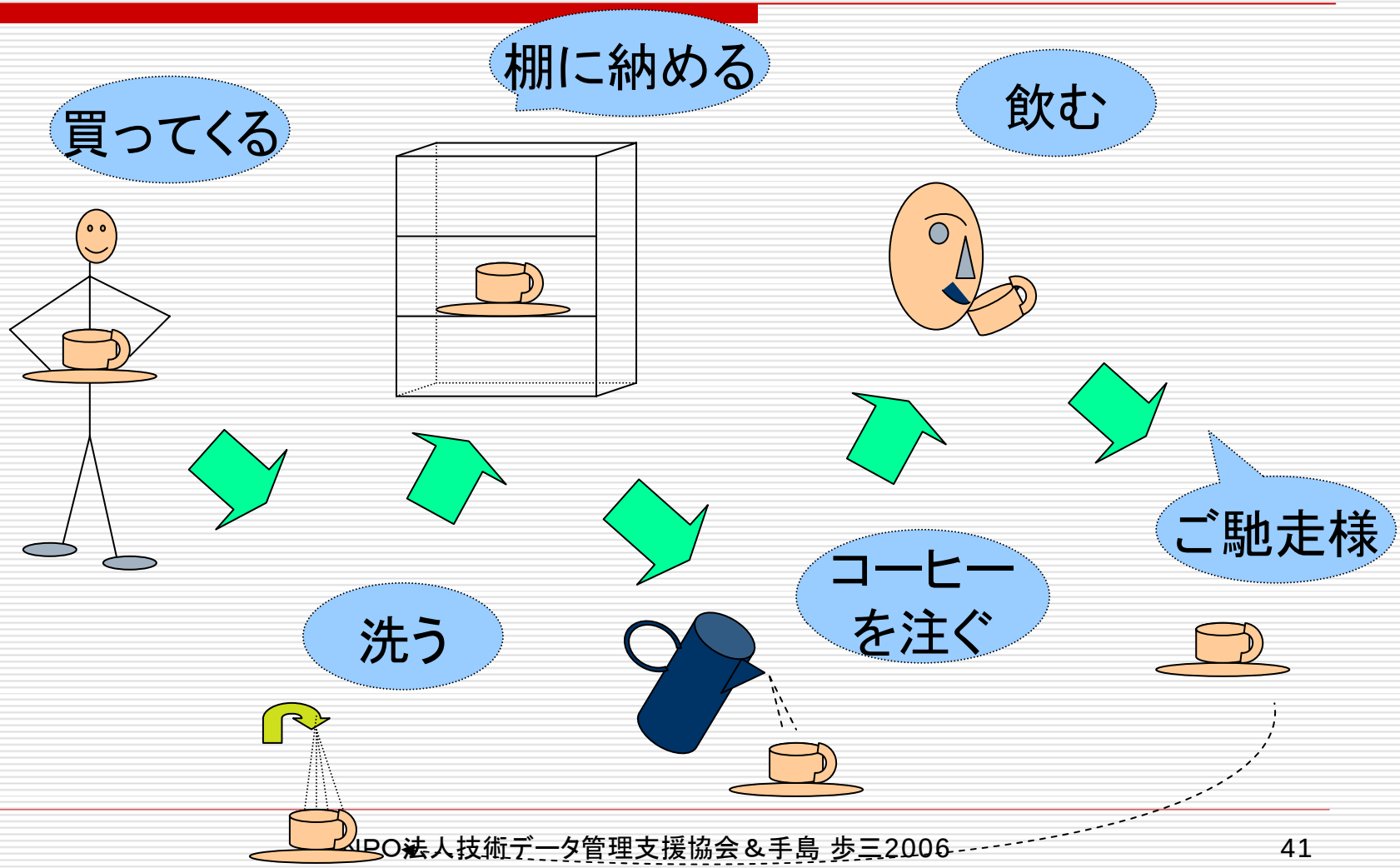
- 実体のライフサイクルを記述する
- 実体の状態を変化させる活動あるいは出来事とそれらの順序（状態遷移規則）
- 活動の識別子と属性
- 実体の特定な状態によって起動させる活動を記述することがある

動的モデルの例



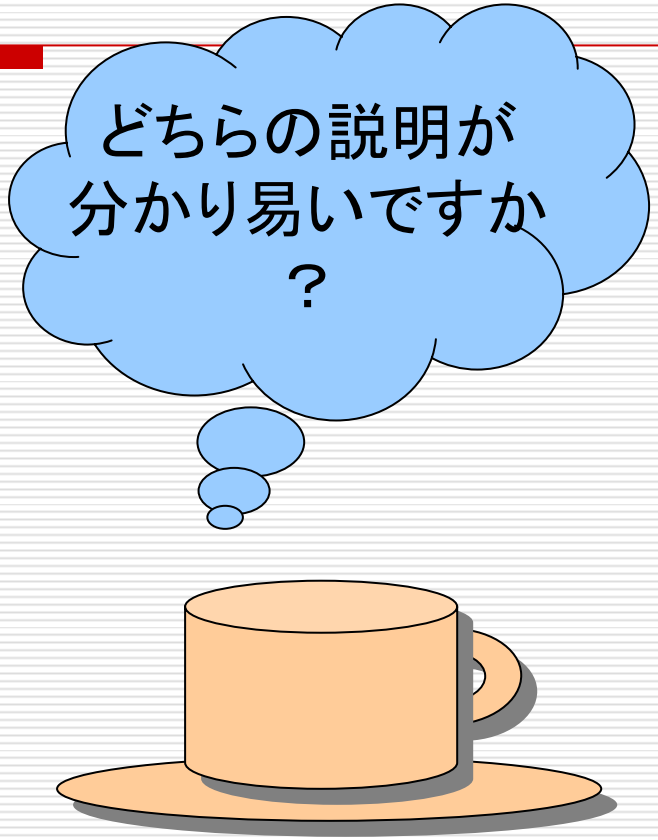
引用：手島ほか「ソフトウエアのダウンサイジング」日本能率協会マネジメントセンター

コーヒーカップの動的性質

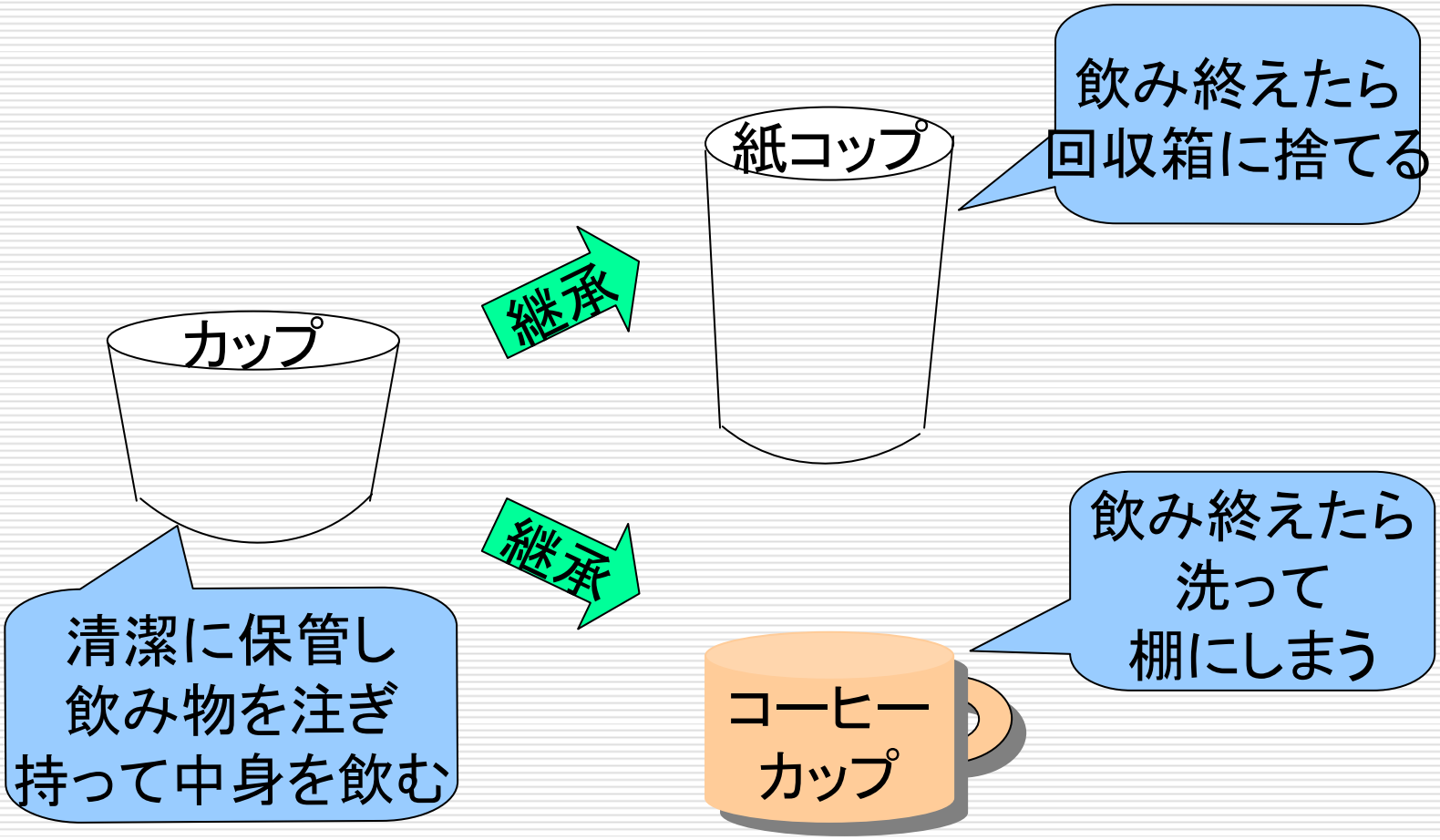


コーヒーカップの説明

- 属性による説明
  - コーヒーカップは直径約8cmの半球形または竹筒を切断した形の容器に取っ手をつけた金属あるいは陶器です。ソーサーと組になっています。
- 動的性質による説明
  - コーヒーカップは店で買ってき洗って棚にしまっておきます。コーヒーを注いでソーサーに載せ、お客様に出します。お客様はカップを持って飲みます。飲み終えると洗います。



クラス階層と継承

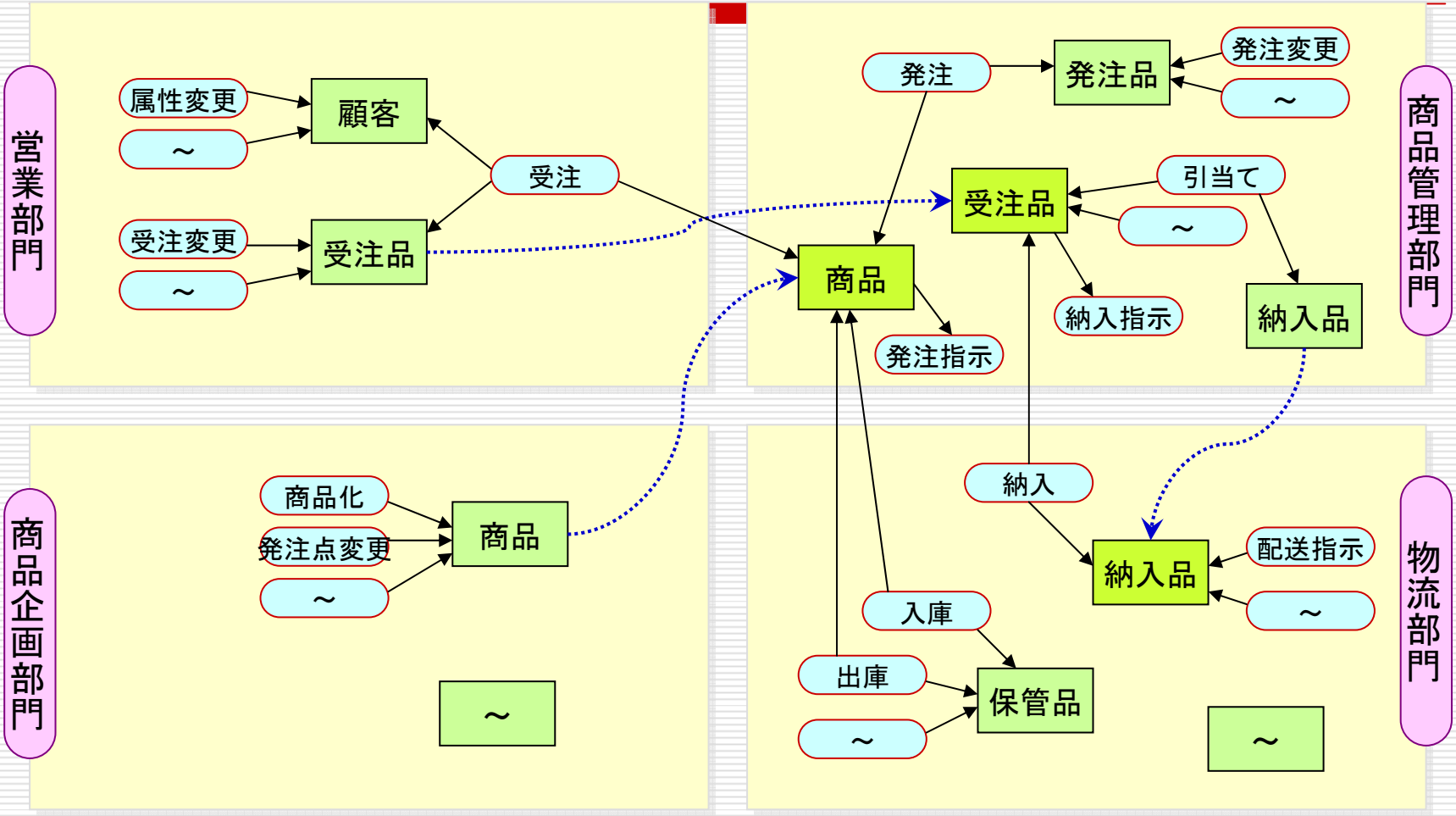


実世界で行なわれる並行処理を可能にする業務規則の簡素化

- 実世界において出来事はランダムに発生しているように見える
  - しかしそこには規則性がある
- 一つの対象を取り上げると、一定の規則にしたがって変化している
  - ある対象に働きかけても、対象がそれを受け入れる状態に達していないなら、その働きかけは不発に終る
  - ある対象が、ある状態に達した時、何らかの活動が起動される、あるいは出来事が起きる

- これまで「情報処理機能」を手続きとして表現してきた。手続きは諸条件の組み合わせにより煩雑となり、論理的な隙間が発生しやすい。
- 個々の「もの」を中心に自然法則やビジネス活動の順序を分解してみると、規則は単純化され、かつ独立性が高くなる。
- 手続きの自由度が高まり、臨機応変の対応も容易になる

組織間連携モデルの例



引用：手島ほか「ソフトウエアのダウンサイジング」日本能率協会マネジメントセンター

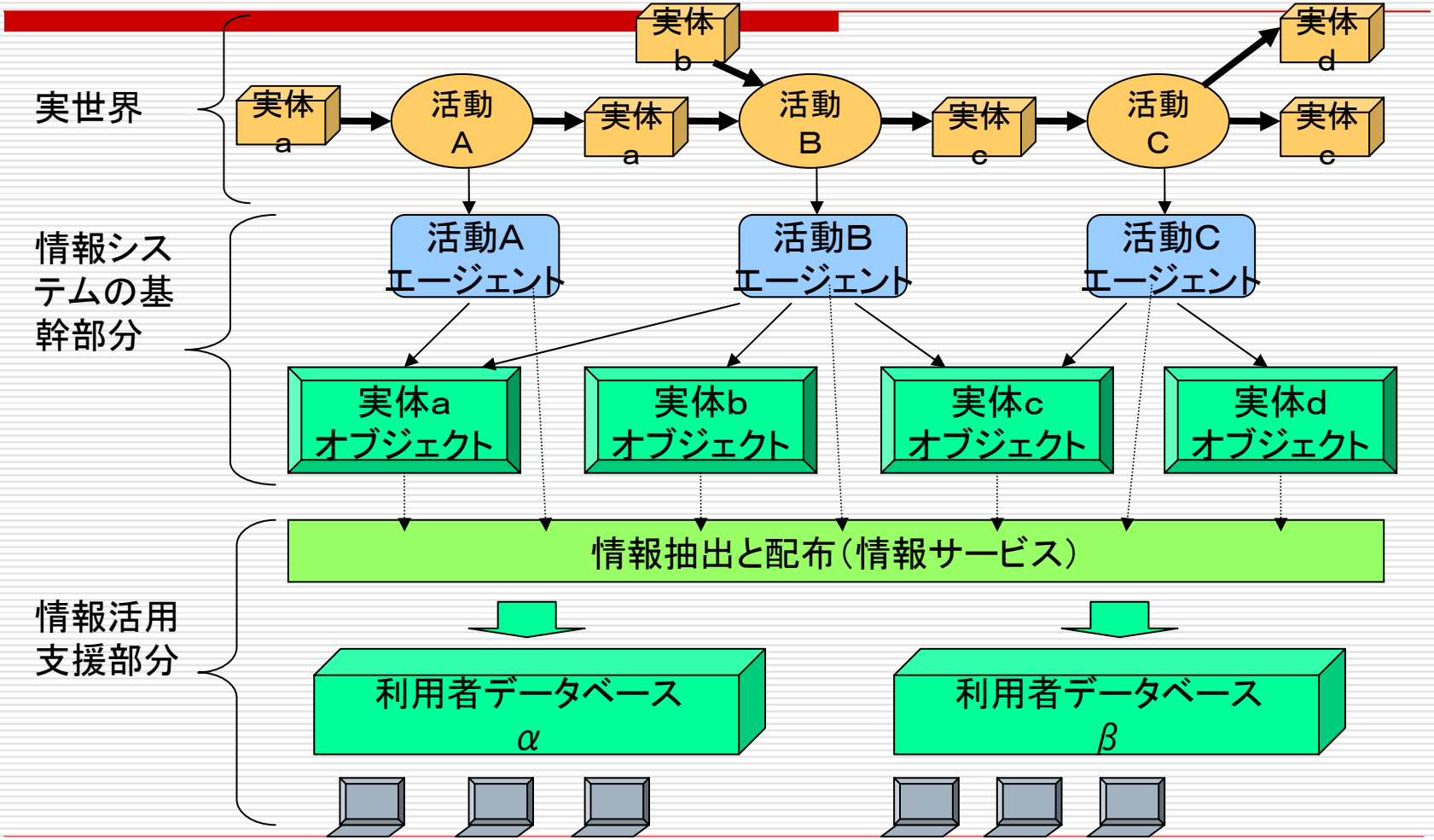
組織間連携モデルに顕れる
組織活動を支援する情報の流れ

- 組織の「もの」と活動に対する責任・権限と組織間連携の姿を記述する
- 「もの」の管理責任と情報の品質保証責任の一致
- 実体を管理責任を持つ機能部門に配置し、実体の状態を変化させる活動を実行部門に配置してその関係を示す
- 実体の管理責任が階層化されていて責任の分担があるなら、実体を分散配置する
- 動的モデルに組織空間の概念を加えたもの
- 活動の現場でトランザクション・データを即刻、ただ一度だけ採取
- データの品質をチェックし、問題があれば出来るだけ速やかに訂正
- 物を管理する責任者の手許に物状態を捉えるデータベースを分散配置
- 活動の現場と物の管理者を繋ぐデータ伝送
- 物の状態に応じて活動を起動する組織的連携

相互作用系に現れる実体種類

- オブジェクト
  - 活動によって状態変化する受け身の存在
  - 特定の状態において何らかの活動の起動を要請することがある
- エージェント
  - オブジェクトに関する知識と行動能力を持ち特定の活動を遂行する
  - 他のエージェントに行動の代行を要請することがある
- アクター／クライアント
  - システムの利用者
  - エージェントに何らかのサービスを要請する
  - 行動規則を持たない・強制できない

再掲
オブジェクト指向情報システムの概念的構造



４．２　情報系アプリケーション導出の考え方

- 情報を取りに行く
  - 利用者の職務権限と能力に相応しい情報を利用者データベースに供給する:「情報サービス」
    - エンドユーザ・コンピューティング用市販ソフトウェア（表計算、データマート／データウェアハウスなど)を利用
  - 利用者が自らの手で欲しい情報を取り出す:「エンドユーザ・コンピューティング」
    - アウトプットデータ作成用のソフトウェアはなるべく作らないで済ませる。
    - 損益計算書やバランスシートなどの定型書類は市販の専用ソフトウェアを利用してもかまわない。
- 定型レポート処理
  - データベースやトランザクションデータのログを加工して、定型データを取り出す仕組を用意する
  - 例：総勘定元帳、バランスシート、損益計算書
- 意思決定支援
  - 素データを市販のデータ加工ツールを用意し、必要に応じて処理する

４．３　アプリケーション・パッケージの選択とカスタマイズ

留意点

- ビジネスの事実を表すデータを取り扱うこと
  - 全てのアプリケーションはビジネスの事実を表すデータのみを取り扱うものでなければならない
- 機能名に惹かれてパッケージを選んではいけない
  - 機能名は名前にすぎない
  - 処理内容に注目する必要がある
- 使わない機能はトラブルの原因になりやすい
  - 間違えて使う
  - 思いがけない副作用

選択基準

- データ構造の一致
  - データ仕様を合わせるチューニング
  - 構造が合わないときは機能を使うことはできない
- 機能の選択
  - 不要な機能を棄てる
- カスタマイズ
  - 不足するデータ種類を補う
  - 不足する機能を補う
  - 補う方法がないパッケージは選ぶべきでない

5. 情報基盤整備

- 情報基盤整備の目標
  - 組織構造に合う情報システム構造の分散と統合を可能にする
  - 業務形態に適合する情報処理形態を選べるようにする
- 情報基盤構想の役割
  - アプリケーションの分散配置方針設定（Targeting）
  - 情報処理効率と品質の保証を可能にする情報技術商品選択
  - 情報基盤技術を理解するソフトウェア業者の選択
- ソフトウェア開発環境整備
  - 処理形態によって異なる情報基盤商品
  - 処理形態によって異なるプログラミング言語

情報基盤整備の要件
基本ソフトウェアとミドルウェアの整合

- OS
- DBMS
- 通信制御
- GUI
- その他の入出力装置制御
- 言語プロセッサ
- エディタ

- DDDS（データ・ディクショナリ ＆ ディレクトリ・システム）／リポジトリ

- インタープリータ類
  - 言語
  - データ・マッピング
  - コード変換
- 運用管理システム
  - 処理手続
  - トラブル処理／リカバリ
  - 課金（処理費用）
- セキュリティ
  - 使用権
  - データ・アクセス権
  - ウイルス対策

6. ビジネス改革と情報システム構築の同期

- 使わないアプリケーションを構築すると
  - 本当に必要なアプリケーションの実現時期が遅れる
  - 投資回収の遅れ
  - 実用までに多数の変更が発生するだけでなく、場合によってはせっかくのアプリケーションが不用になる
  - 無理してアプリケーションを使用すると、ビジネスに予定外の歪みが生じ、ビジネス改革シナリオが狂う
- ビジネス改革との同期
  - 急ぐ、明確なことから、小さく、迅速に実現する
  - 情報システムを独立性が高い、短期間で実現できる構成要素に分解する
  - ビジネス改革シナリオに沿う、優先順位に基づいて小さな改革課題を取り上げる
  - そのビジネス改革課題を可能にするための情報システム構成要素を割り付ける
  - 移行とテストの計画を補う

ソフトウェアJITの手順

- ビジネスモデル改革、サプライチェーン整備を目指すビジネスプロセス改革構想
- 業務改革のシナリオとプログラム
- 業務改革を可能にするイネーブラー提供スケジュール
- 情報と情報技術提供の計画
- 移行計画
  - 共通データ管理基盤整備
  - データ取り込み
  - 周囲のアプリケーションとの結合
- テスト計画
  - テストデータ(インプット＆アウトプット)作成
- 進化型プロトタイピング
  - プログラムの骨格をデータ構造に基づいて導き出す
  - モジュール内容
  - ユーザインターフェース
  - データベースが正しく生成されることの確認を優先
  - コードレビュー：不用な機能が付いていないことのチェック
  - プログラムができあがって後にユーザに実際に使ってもらい、「要求」を引出す