システム開発（まとめ）

(情報システム開発論、第１４回講義)

Email　fwhy6454@mb.infoweb.ne.jp

URL　http://homepage3.nifty.com/suetsuguf/

参照　情報と社会（http://homepage3.nifty.com/suetsuguf/johou0tx.htm ）

作成者　末次　文雄

ソフトウェア工学

- 発想の原点
  - ソフトウェア開発・製造において、
  - 徒弟的な経験と勘で進めるのではなくて、
  - 従来の工学分野と同様に
  - 有効な手法を見出したい

- Software　engineeringの直訳
  - 1968年、NATOの国際会議で出た用語

ソフトウェア工学の目的

- 情報システム、およびソフトウェア製品の
  - ・製作技術を確立し
  - ・要求仕様を満足し

  （システム設計・開発）

  当科目
  - ・正しいと証明ができ
  - ・予定された納期
  - ・予定の予算内で

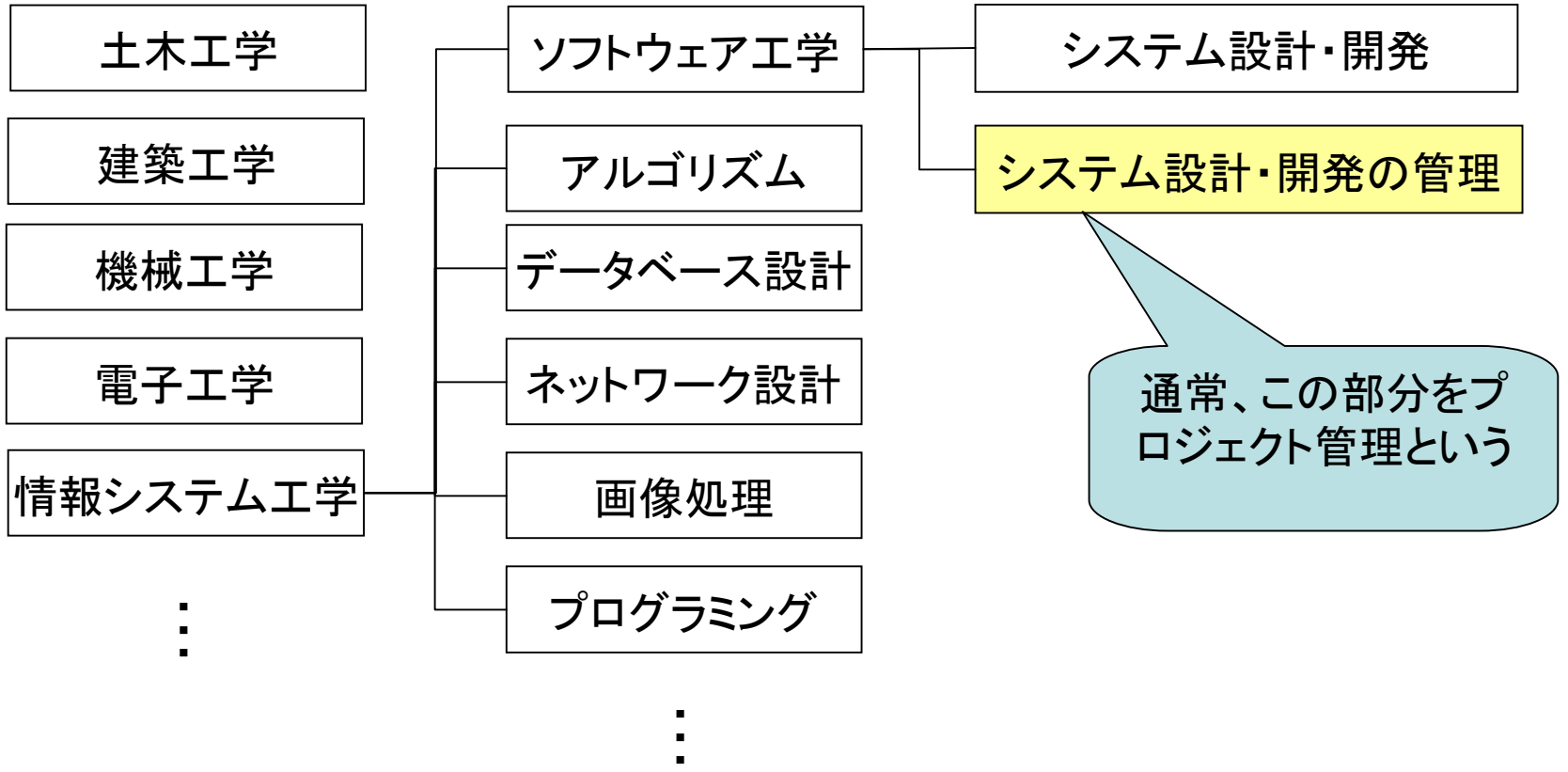
  （プロジェクト管理）

  完成できることを支援する

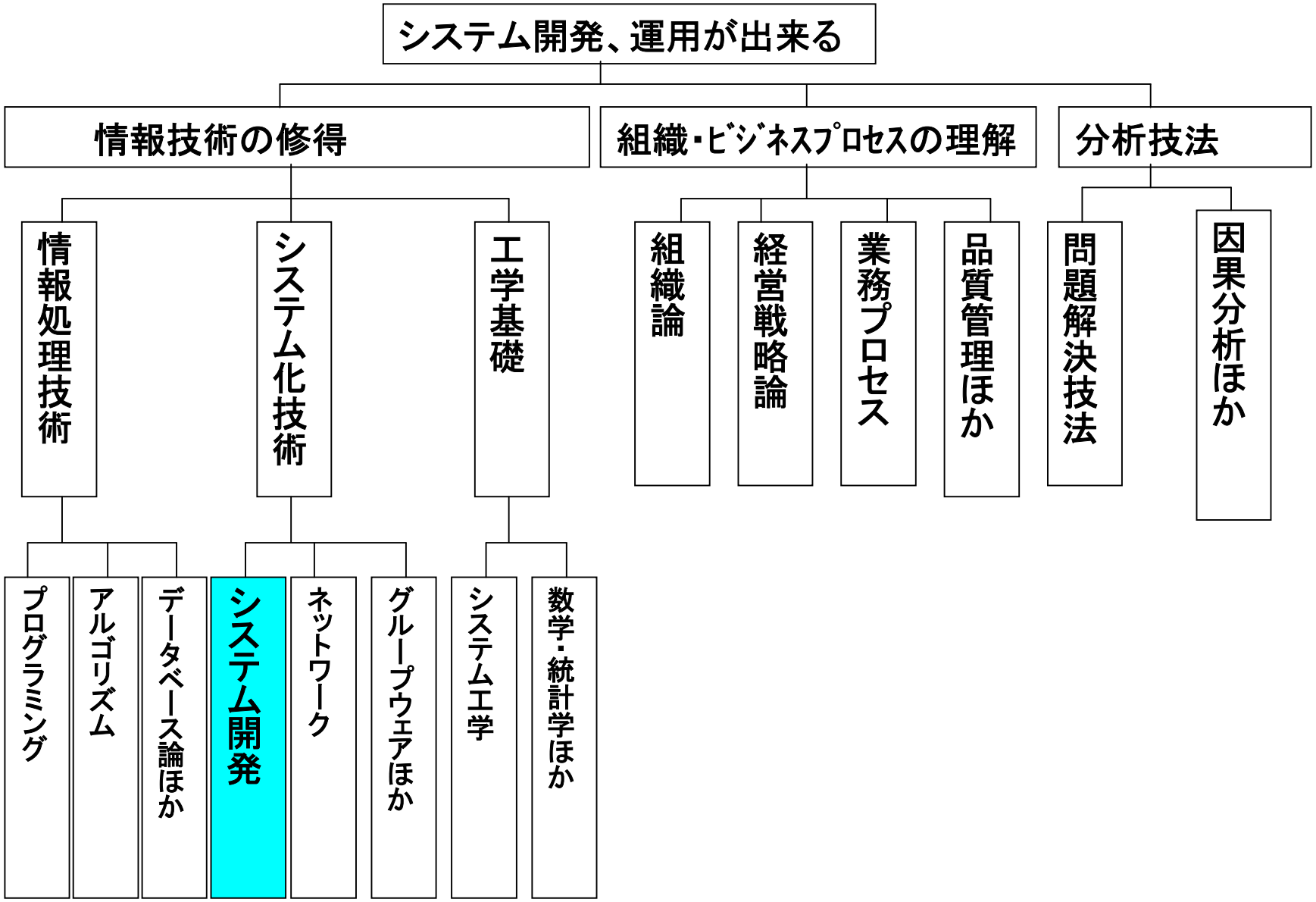
プロジェクト管理の位置づけ

- ここで言うプロジェクト管理とは、

  「ソフトウェア開発・製造を実行するプロジェクトの管理」



復習：知的情報システム工学科における当科目の位置付け



受講ガイダンス

科目内容と修得目標

- 目標　：　この科目を受講することにより、以下の事柄が出来る。

  ①情報システム開発の手順を理解し、簡単に説明する。

  ②情報システム開発の方法を2種類以上理解し、説明する。

  ③UMLを使用して、簡単な情報システムの設計が出来る。

  ④対象業種や業務に対応したシステム設計が出来る。

- 成績評価方法　：　毎回レポートを課す。

  出席回数、レポート、期末試験による総合評価をする。

「情報システム開発論」の内容

開発手法	第1回	:	システム開発の概要
	第2回	:	情報システム開発の流れ
	第3回	:	情報システム開発の手法1 （ﾃﾞｰﾀ中心ｱﾌﾟﾛｰﾁ）
	第4回	:	情報システム開発の手法2 （オブジェクト指向）
	第5回	:	情報システム開発でのUMLの活用
業務のモデル化	第6回	:	経理業務のモデル化
	第7回	:	小売業のモデル化
	第8回	:	卸売業のモデル化
	第9回	:	e-businessのモデル化
実習	第10回	:	簡単な情報システムの設計（課題1：図書館システム）
	第11回	:	同上のシステム設計解答例
	第12回	:	簡単な情報システムの設計（課題2：ネットショップ）
	第13回	:	同上の解答例の発表
	第14回	:	まとめ、課題の発表

プロジェクト管理

システム開発のための科目紹介

システム設計・開発

情報システム開発論（当科目）

・要件定義、外部設計

情報システム開発技術（3年、後期）

・内部設計、プログラム設計

情報システム化技術実験演習（4年、前期）

・要件定義～プログラム開発、テスト

データベース設計・開発

データベース論Ⅰ（2年、後期）

・リレーショナルDBの設計、開発

データベース論Ⅱ（3年、後期）

・オブジェクト指向DBの設計、開発

情報システム化技術演習（3年、後期）

・リレーショナルDBの設計技法

情報システム開発に必要なスキル

技術・知識

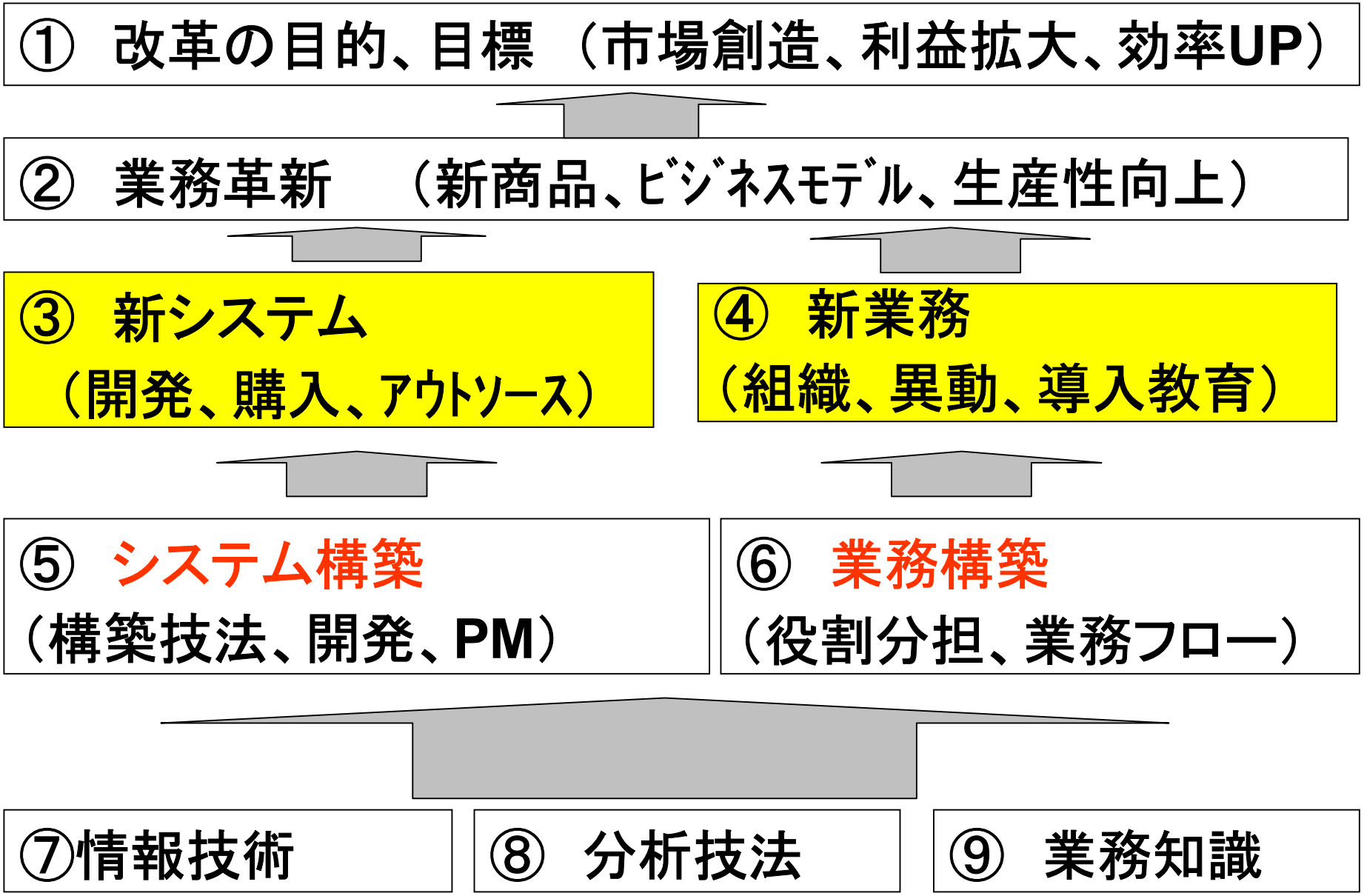
- ITスキル（情報処理技術、開発技術）
- 管理技法（分析技術、コミュニケーション技法）
- 業務知識
  - 主活動　（開発、購入、製造、物流、販売）
  - 支援業務（経営、財務、人事、システム）

SEに必要なスキル（経験値）

- SEの仕事の進め方
  - ー何事も計画を立ててから進める
  - ー書き出してみる（リスト、関連図）
  - ー自分の考えを話し、人からアイデアをもらう
  - ー大きな課題からかたずける
  - ー決まらないことはﾕｰｻﾞｰﾄｯﾌﾟと直談判
  - ー5W3H（What, Why, Who、When、Where, How to, How much, How many）

- SEの心構え
  - ー好奇心→もっと知る→対象業務を好きになる
  - ー忍耐力、持久力、スタートしなければゴールに着かない
  - ープラス思考（入って来る全ての情報を「快」と捉えるクセ）

情報システムの位置付け



企業システムの重点

～1980年代
- 業務の効率化、生産性向上
  - 納期短縮
  - 品質向上
  - コスト削減

1企業に1、2台

→
- 効果大の基幹業務のシステム化
  （開発、生産、販売、購買、物流）
  （各システムは独立傾向）

1990年代
- 変化への対応力強化
  - 顧客ニーズへのすばやい対応
  - 部門間の連携強化
  - 海外シフトへの対応
  - システム費用の削減

1部門に1台

→
- 商品開発期間短縮の支援
  （製品情報のデジタル化、PDM）
- 各システムの連携、統合化（SCM）
- 情報の共有化（OA、ERP、ポータル化）
- ダウンサイジング、NW強化
- 先進IT技術の短期適用（アウトソーシング）

2000年代
- 業績拡大への直接的な貢献
  - 顧客の確保、利益拡大、技術革新
  - ノウハウの継承・蓄積・活用

1人に1台

→
- インターネット技術の活用（販売）
- 顧客対応の強化（CRM、DWH）
- 新技術開拓の支援（仮想実験など）
- 知識・知恵のデータベース化

企業システムの構成

① 複数のコンピュータを用途別に使い分け

- 大型コンピュータ（基幹系システム、データベース）
- スーパーコンピュータ（複雑な科学技術計算用）
- 中型コンピュータ（部門システム）
- 専用コンピュータ（工場システム、物流システム）
- ワークステーション（開発部門の技術者用）
- パソコン（オフィスでは一人一台に設置）

② 全てのコンピュータをネットワークに接続

- 出先、取引先とは専用線で接続（VPN）
- 海外の出先、工場、取引先とも専用線で接続
- 顧客、消費者、社会とは、インターネットで接続

情報システム開発とは

- システムが対象とするもの
  - 人、組織の活動------当科目の対象
  - 機械の制御----------電子/機械工学との境界

- 人、組織の活動（例）
  - 主活動　（開発、購入、製造、物流、販売）
  - 支援業務（経営、財務、人事、システム）

- Make／　**Buy**　／　アウトソーシング

- ユーザー部門との共同作業

開発方法の違い

1. 設計手法に着目

データ中心	・データの流れと分析に重点をおいて進める。(DFD、ER図、データ正規化、構造化設計)	主流。DBは今後も主流
ｵﾌﾞｼﾞｪｸﾄ指向	・属性と処理(＝ﾃﾞｰﾀと操作)を一体化して設計(ｶﾌﾟｾﾙ化) ・ｿﾌﾄの部品化を進め易い。	GUIから始まり、適用拡大

2. 開発プロセスに着目

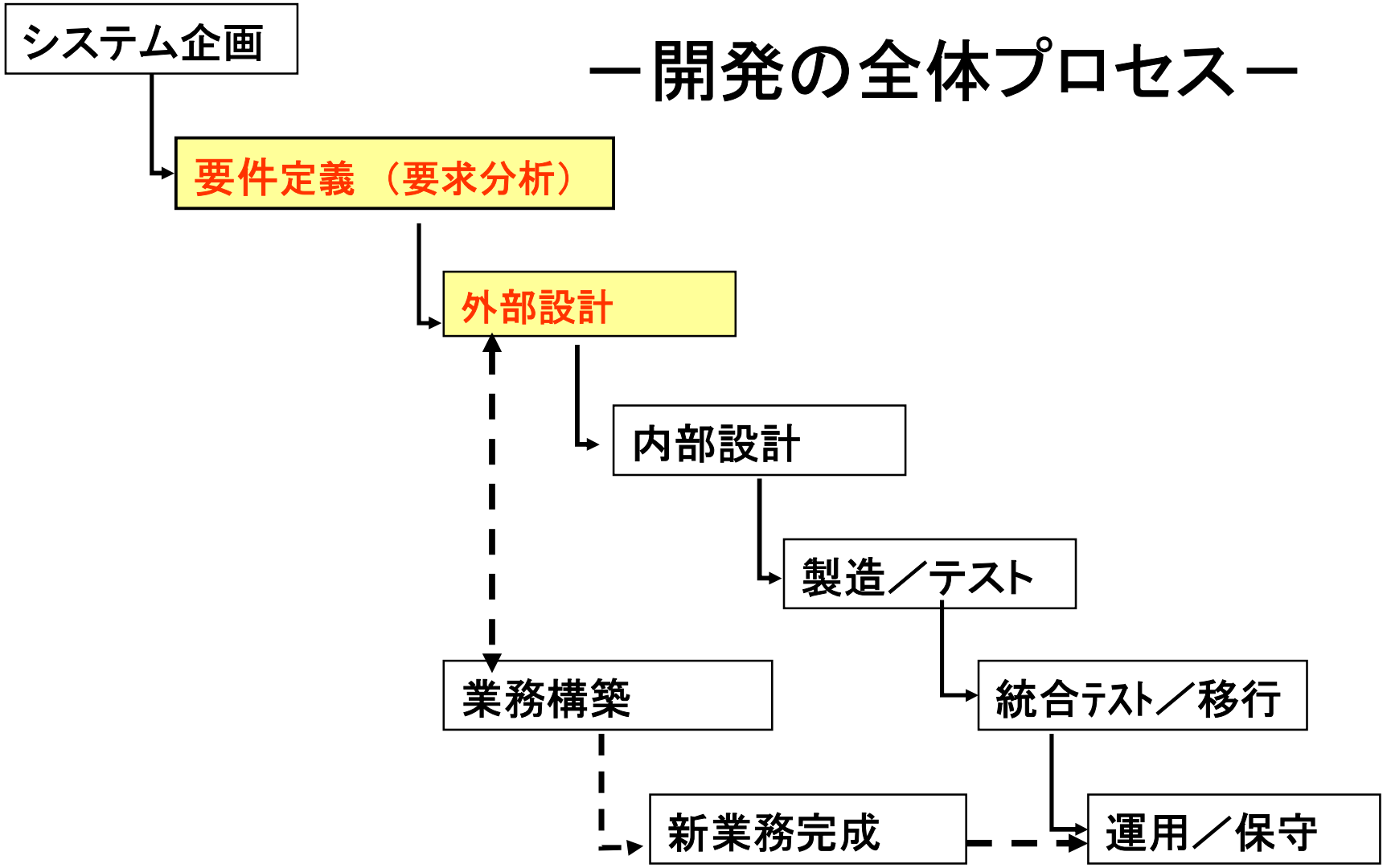
ｳｫｰﾀｰﾌｫｰﾙ方式	・上流から下流までを順を追って開発する方式。(主流)	ﾚﾋﾞｭｰが必須
ﾌﾟﾛﾄﾀｲﾋﾟﾝｸﾞ方式	・部分から試作し、実地検証しながら、繰り返す。(ｽﾊﾟｲﾗﾙ)	工程内での適用

その他の開発方法の特徴

３．実装の配置	・ホスト中心 ・２層、３層	（DB管理、処理、ユーザーインターフェース）
４．規模	・一括開発（中小規模） ・段階開発（大規模）	（フェーズ分け）
５．新規／改造	・新規開発 ・改造型開発	
６．システム特性	・リアルタイム処理 ・バッチ処理	
７．企業統合	・一社のシステムへ統合 ・ブリッジ方式 ・新規システム開発	（S/W、 H／W）
８．使用技術	・低水準、高水準言語 ・オブジェクト指向言語	

開発プロセス

－開発の全体プロセス－



（ 部分が当科目の主な対象）

システム開発工程の概要

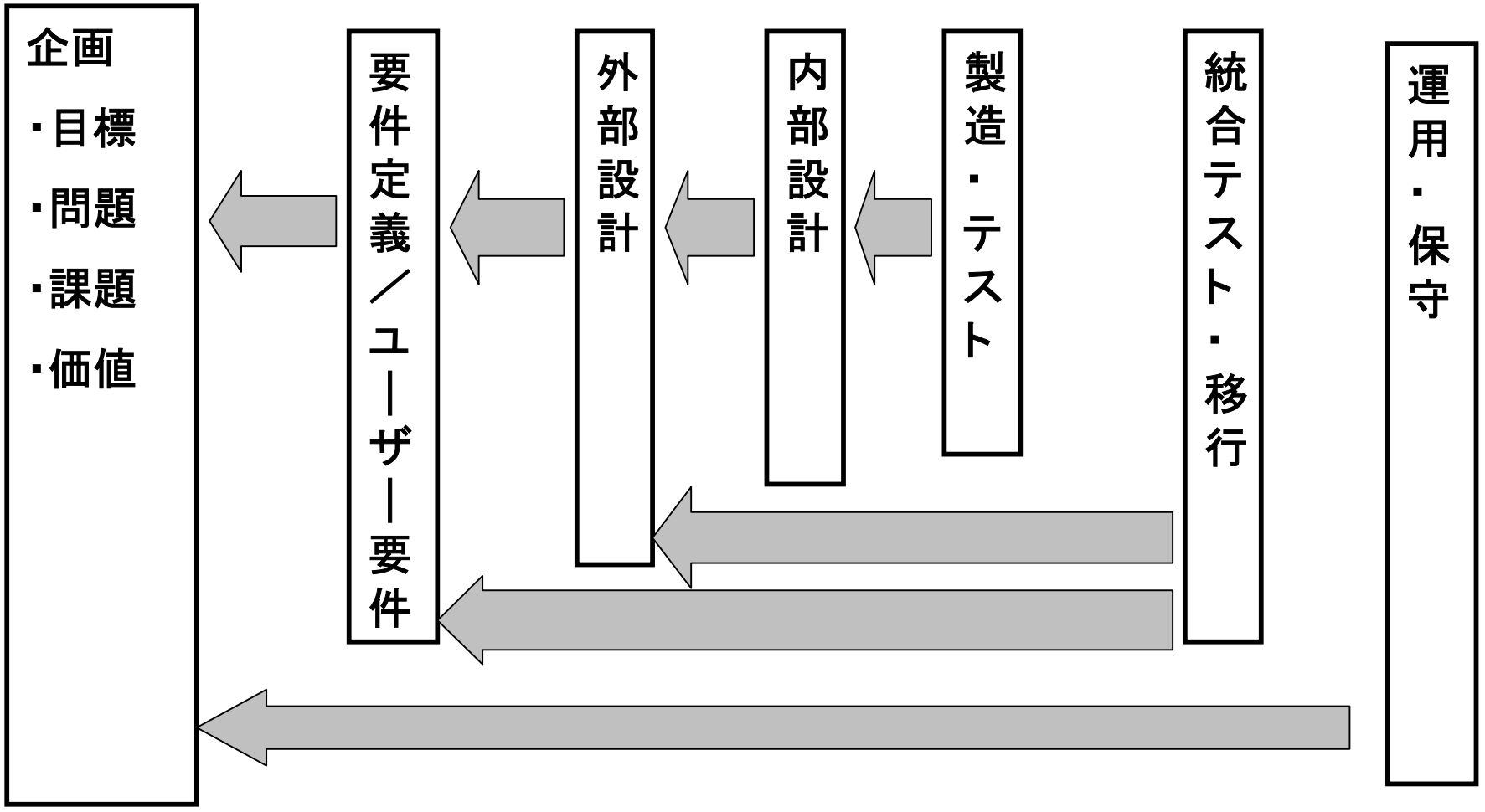
	仕事の内容	OO方式
システム企画	・目的、方針、ﾋﾞｼﾞﾈｽﾓﾃﾞﾙ、機能、構造、効果、予算	同じ
要件定義 （要求分析）	・要求の調査／分析、範囲、要求仕様まとめ（機能・DB・品質）、実現性、費用、計画作成	要求ﾓﾃﾞﾙ
外部設計	・機能、DB、I/O、構造を決定	分析ﾓﾃﾞﾙ
内部設計	・上記の物理ﾓﾃﾞﾙ（実装ﾚﾍﾞﾙ）	設計ﾓﾃﾞﾙ
製造／テスト	・ﾌﾟﾛｸﾞﾗﾑ製造、テスト	同じ
統合ﾃｽﾄ／移行	・本番並みのﾃｽﾄ、ユーザー承認、本番移行	同じ
運用／保守	・運転、監視、保守（改善・バグ修正・ﾄﾗﾌﾞﾙ予防）	同じ

開発プロセスのポイント

	工程のポイント
システム企画	・何のためにどういうシステムが必要かを提案し承認を得る
要件定義 （要求分析）	・具体的に何がやりたいかをまとめて、かつ実現可能性を検証する
外部設計	・ユーザーの立場に立って、必要な仕様を決める（＝ユーザーマニュアルの完成に等しい）
内部設計	・実装ﾚﾍﾞﾙの仕様を全て決定
製造／テスト	・上記に基づいて、実装する
統合ﾃｽﾄ／移行	・本番並みのﾃｽﾄ、ﾕｰｻﾞｰ承認
運用／保守	・運転し、かつシステム育成

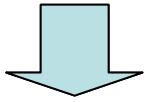
各工程の対象

各工程は、矢印で示す上流工程の各要件を、満足している必要がある。

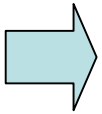


プロジェクト管理の必要性

- 情報システム開発がもつ潜在的な問題点
  - ・歴史が浅く、製造技術が未確立
  - ・従って見積り技術も明確でない
  - ・仕様書も、人によるバラツキが非常に大きい
  - ・製品が目に見えず、出来上がりが分かりにくい
  - ・組織、人間の活動を対象としており
    複雑で、実現手段が複数あり
    しかも頻繁に変更される

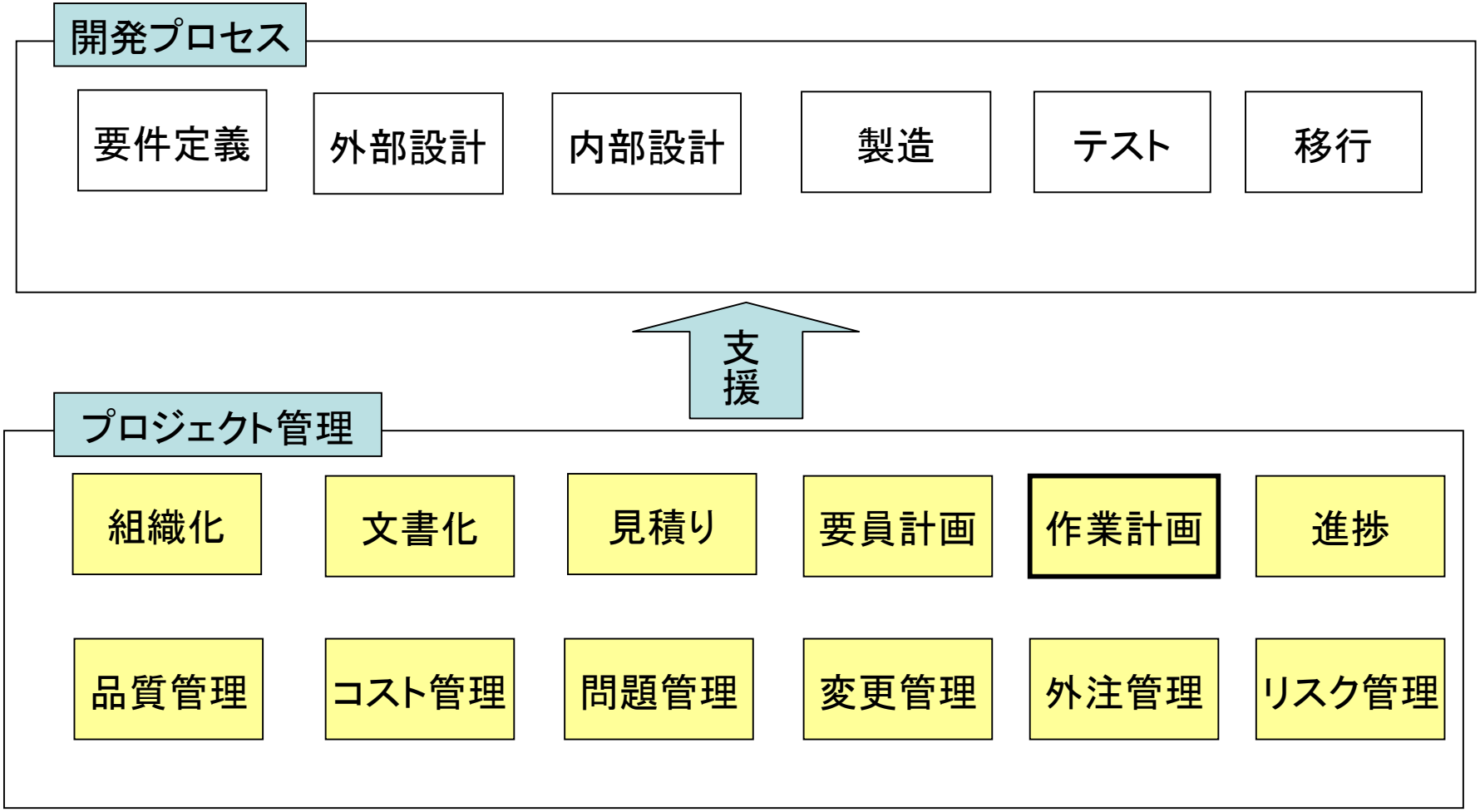


・納期、品質、コストの未達が
当然のように起きる



・開発プロセスの改善だけ
では不足であり、管理が必須

プロジェクト管理の構成



プロジェクトマネージャーの心得

- 組織化
  - 高度な技術保有者は、先行して確保（もともと少ない）
  - メンバーが出来ないという言い訳を全て解決
  - メンバーの本音を聞ける人間関係
  - メンバーへの指示は具体的にする
- 文書化
  - システム設計・開発は文書作成の連続である
  - 要求分析時、外部設計時に考えた内部仕様は文書化しておく
  - 何故そうしたかのねらい・理由は必ず書かせる
- 要員計画
  - メンバーの教育時間を確保する
  - 表面的な理解はケガのもと
  - 外部仕様から内部仕様に変換するのは、もともと高度な作業だ
- 作業計画
  - 作業開始の前には、用語集を準備する
  - プログラム製造に先行してソフト部品を揃える
  - 統合テストは、外部仕様書を元に作成すること
  - どうしても出来ないことはやらない
- 進捗
  - 週次報告書には全て目を通す
- 問題管理
  - 問題点は、目標達成の一里塚
- リスク管理
  - いつも最悪のリスクを想定し、その答えを持つ

ﾃﾞｰﾀ中心型ｱﾌﾟﾛｰﾁの考え方

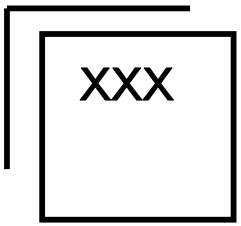
- 業務処理の流れをデータの流れとして把握し処理は単にデータの変換に他ならないものと考える。
- データ項目は永続性がある。（処理は変わる）
- それによって、システム開発の上流工程での品質確保、開発効率の向上をねらった手法。

DOA開発の手法

①DFD （Data Flow Diagram）

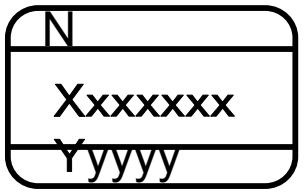
- 適用業務におけるデータの流れを
  ４つの単純な記号で表記
- さらに処理内容を3つの観点で、記述
  - データフロー記述（ﾃﾞｰﾀﾌﾛｰを流れるデータ項目の記述）
  - データストア記述（データストアのﾃﾞｰﾀ項目の記述）
  - 処理機能記述（各ﾌﾟﾛｾｽの処理内容を記述）
- トップダウンで詳細化

DFDの表記法

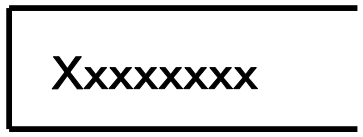


－－－データの発生源／行き先（External　Entity）
・組織、人、サブシステムなど

－－－データの流れ（Data　Flow）
・データフロー名を記入（物の流れ、処理は不可）



－－－データの処理、変換（Data　Process）
・処理名称
・Nは識別番号、Yは担当部門名など



－－－データ・ストア（Data　Store）
・台帳、データベース、ファイルなどデータ蓄積

DFD記述書の作成

データフローの分析結果を明確にするために、
3種類の記述書を作成する。（＝構造化仕様という）

・データフロー記述－－－→画面／帳票設計に使う
　　－データ項目、意味、桁数など
・データストア記述－－－→データベース設計に使う
　　－データストアに含まれるデータ項目の記述
　　－先でデータベース仕様書として完成
・処理機能記述－－－－→プログラム設計に使う
　　－処理ボックスの処理内容を記述
　　－通常、最下位レベルの処理ボックスが対象

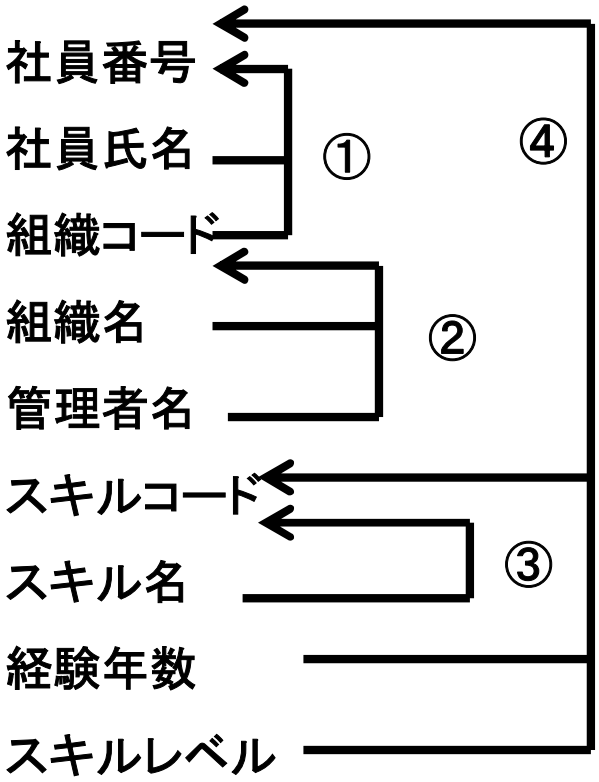
②データの正規化 （Data Normalization）

- DB設計に先立って、
  関連性の強いデータ項目を
  正規化理論に基づいて
  グルーピングする
- その結果、ダブリなどの冗長さを排除
- 非正規形（Unnormal Form）から、
  正規形（Normal Form）を作成する。

データ正規化の簡便な方法

・データ項目間の従属性に着目し、
・全てのデータ項目間の従属関係を洗出す
・従属関係にあるデータ項目のグループを
　独立させる。

社員スキル管理表＝



①社員＝社員番号＋社員氏名＋組織コード

②組織＝組織コード＋組織名＋管理者名

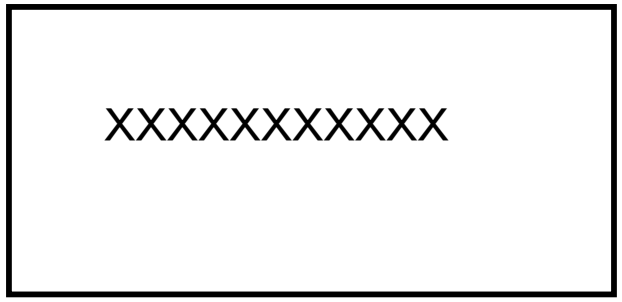
③スキル＝スキルコード＋スキル名

④保有スキル＝社員番号＋スキルコード＋
経験年数＋スキルレベル

③ER図　（Entity-Relationship　Diagram）

- 適用業務が管理すべき対象をEntityとし、Entity間の関連を簡単な記号で定義する。
  - Entity、Entity間の関連（Relationship）は属性（データ項目）をもつ。
- 二つの作成方法がある。
  - DFDのデータストアの正規化結果から作成
  - 対象業務のあるべき姿から作成

ER図の表記法



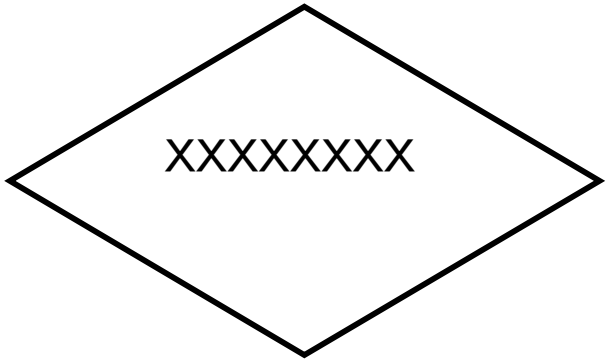
---エンティティ（実体）

・管理対象となるもの

・DBの候補

・厳密には1件づつをエンティティと
いい、集合はエンティティセットという

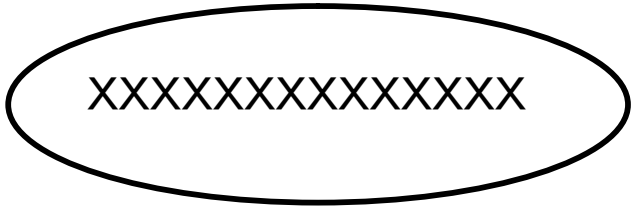
通常、エンティティと略称



---関連

・エンティティ同士の間にある。

・これもDBの候補



---属性（データ項目）

・エンティティ、および関連に含まれる
データ項目である

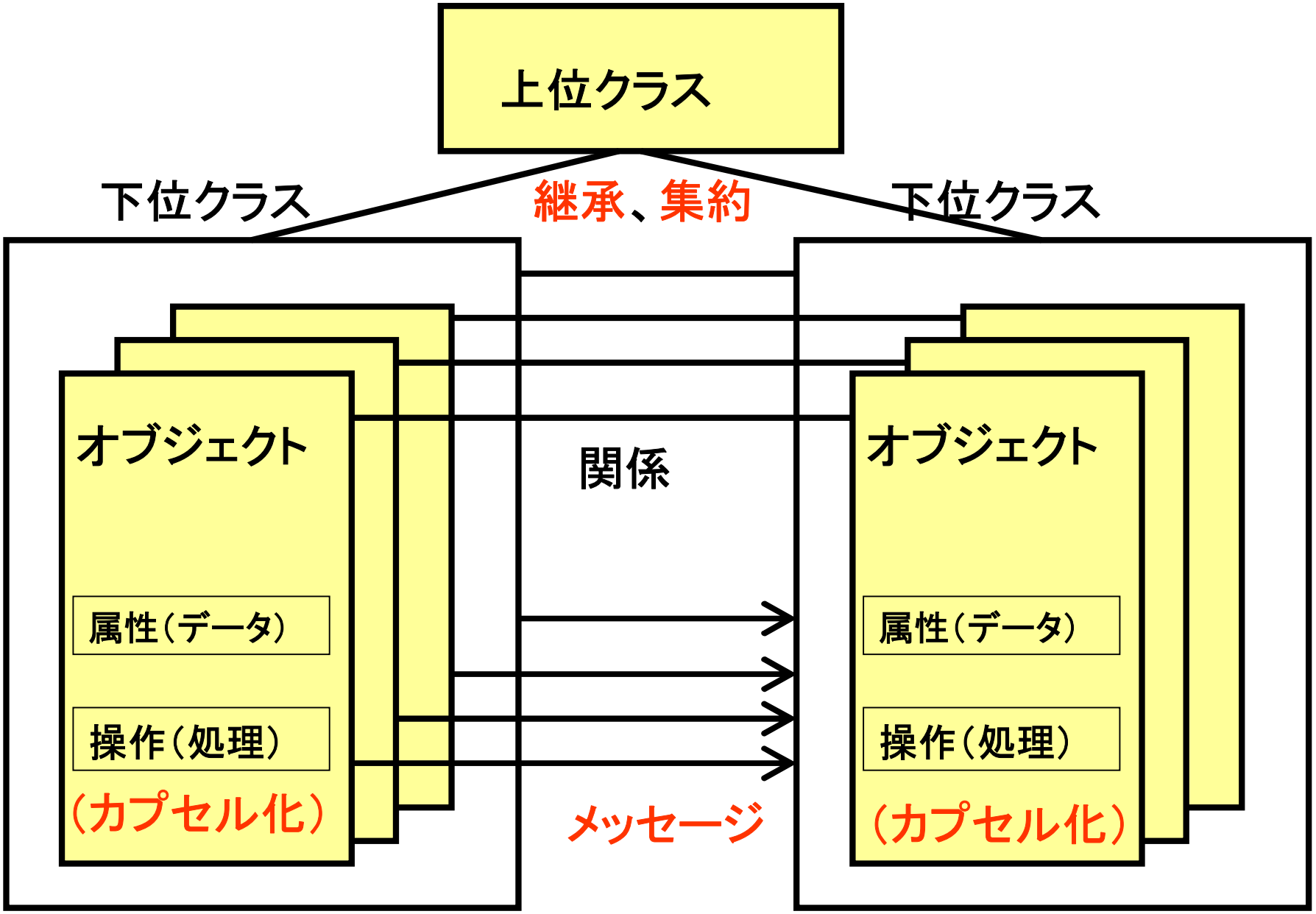
オブジェクト指向の特徴

発想の原点

- 現実の世界は、いろいろなモノが役割を分担しながら機能を果たしている。
- 自分でできることは、自分でやり、自分の範囲外の仕事は人に任せて、結果だけをもらう

それをシステム作りの発想にしたものである。
「現実のモノ（オブジェクト）およびモノ同士の関係を、
そのままソフトウェアで表現することによって、
現実世界の仕組みを、
コンピュータ上で再現するもの」

オブジェクト指向の全体像



オブジェクト指向の基本概念

①オブジェクト、クラス、インスタンス

・オブジェクトは、特定のアプリケーションにおいて、一つ一つを識別すべき対象であり、

・実際に存在するモノ、概念

・その単位は、ERモデルでいうエンティティと近似

ー人　　：社員、顧客、株主、学生、先生・・・
ー組織　：自社組織、特約店・・・
ー場所　：本店、支店、販売地区、工場、倉庫、
　　店舗、配送拠点・・・
ー物　　：商品、製品、部品、材料、設備、機器、
　　金、伝票、帳簿・・・
ー事象　：受注、発注、入庫、入金、売掛・・・
ー概念　：業務、スキル、目標、法規・・・

- クラスは、類似性の高いオブジェクトから共通の特性を抜き出して、定義したもの。

- いわば、オブジェクトの集合であり、抽象化したもの。（実際には、クラスからオブジェクトが生成されるという関係である。）

抽象化とは、共通性を抜き出して分かりやすい概念とすること。

- したがって、クラスは、オブジェクトと同様の特性をもつ

（属性＋操作＋他クラスとの関連）

- インスタンスは、プログラムの中で使う用語であり、オブジェクトと同義。

②情報隠蔽（カプセル化）、メッセージ

- オブジェクト指向では、オブジェクトが互いにメッセージをやりとりすることによって、システムの機能を実現する。
  （ 通常は、オブジェクトの操作名をメッセージとする。）

・オブジェクトの利用者は、このメッセージを送って属性の検索、操作の実行を依頼するしか許されていない。
これをカプセル化という。

ー情報の隠蔽ー属性の隠蔽
ー操作の隠蔽
ーﾒｯｾｰｼﾞﾄﾞﾘﾌﾞﾝ

} カプセル化

・部品化、再利用化が容易になる。

③　継承、汎化、特化

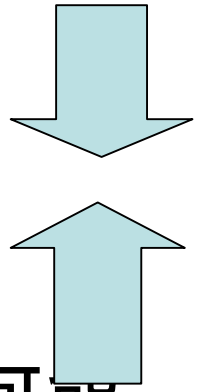
- 対象を分析して、上位クラスと下位クラスに分割することができる。（ｽｰﾊﾟｸﾗｽ、ｻﾌﾞｸﾗｽ）

・このとき、下位クラスは、上位クラスの特性を引継ぐことができる。
（継承（インヘリタンス）という）
（is－a関係、a－kind－of関係ともいう）

・下位クラスは、必要な属性、操作を追加するのみ。

・特化－－クラスを詳細化する過程で、新に下位クラスを導き出すこと。

・汎化－－定義済みのクラスの共通的な特性に着目し、共通部分を取り出し、上位クラスとして導き出すこと。

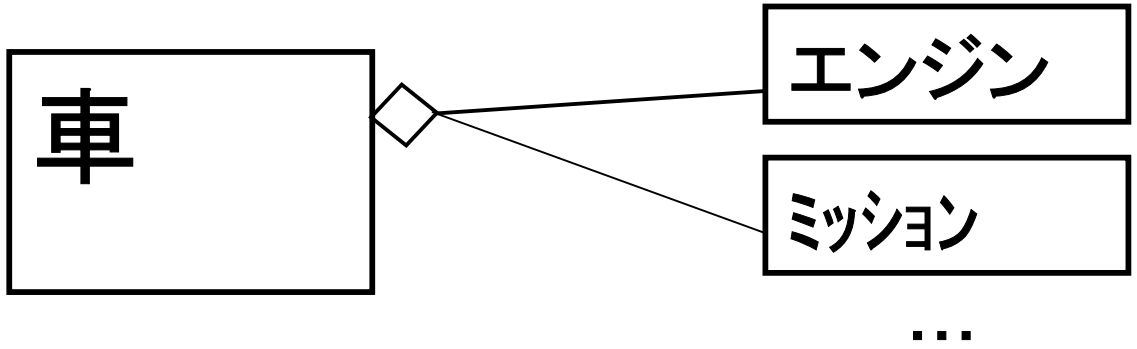


・変更に対して柔軟、重複部分の開発を削減可能。

④ 集約

- オブジェクトが他のオブジェクトを包含しているときに、二つのオブジェクトは、集約の関係にあるという。
  （所有の関係ともいう）
  （has－a、part-of関係ともいう）
- これらは、複合ｵﾌﾞｼﾞｪｸﾄ、ｺﾝﾎﾟｰﾈﾝﾄ･ｵﾌﾞｼﾞｪｸﾄともいう。
- クラスが複雑な場合は、この集約の概念で、シンプルなクラスに分解することが出来る。

（例示）



UMLの分類と用途

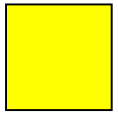
			ダイアグラム	役割
機能モデル	機能構造	1	ユースケース図	システムの利用者から見た、外部機能、その順序を定義する。あわせてシステムの境界を定義。
論理モデル	静的な構造 （システム構造） （概念モデル）　（オブジェクトモデル）	2	クラス図	概念や静的なクラス間相互関係を表現し、 モデルの構成部品の集まりである。
		3	オブジェクト図	システムのある時点でのオブジェクト状態のスナップショット。
		4	パッケージ図	モデル要素の階層的グルーピングであり、モデルの構成を管理する。（クラス図の特殊形）
	動的な構造 （振る舞い、状態）	5	シーケンス図(相互作用図)	オブジェクト間のメッセージ交換の時系列表現
		6	コラボレーション図（相互作用図）	オブジェクトの相互作用を直接に表現する。
		7	状態図（ステートチャート図）	オブジェクトの取りうる状態，遷移などライフサイクルを表現する。
		8	アクティビティ図	システムの動作の流れの表現。状態図の一種
実装モデル	物理的な構造	9	コンポーネント図	システムを構成する実行可能モジュールやソースコードの構造を表現
		10	配置図	システムを構成するマシンや装置（CPU、デバイス等）の分散配置を表現

開発工程とUML

以下に、どの開発工程で、どのUMLダイヤグラムを使用するかを示す。

開発工程	使用するUMLダイヤグラム
要件定義	・ユースケース図、および記述書
外部設計 （分析フェーズ）	・クラス図、オブジェクト図、パッケージ図 ・シーケンス図 ・必要に応じて、コラボレーション図、状態図、アクティビティ図 ・ユースケース図の詳細化、クラス仕様
内部設計 （設計フェーズ）	・コンポーネント図 ・配置図 ・追加項目をクラス図、シーケンス図等に反映。クラス仕様書。

UMLの用途



- 一般的な概念の説明
  - クラス図　　（体系、分類）
- 組織・業務の分析
  - ユースケース図　（業務機能）
  - クラス図　　（管理対象の明確化）
  - シーケンス図　　（業務の流れ）
- システム開発
  - ユースケース図　（システム機能）
  - クラス図　　（システム機能構造）
    （データモデル構造）
  - シーケンス図　　（動的システム機能構造）

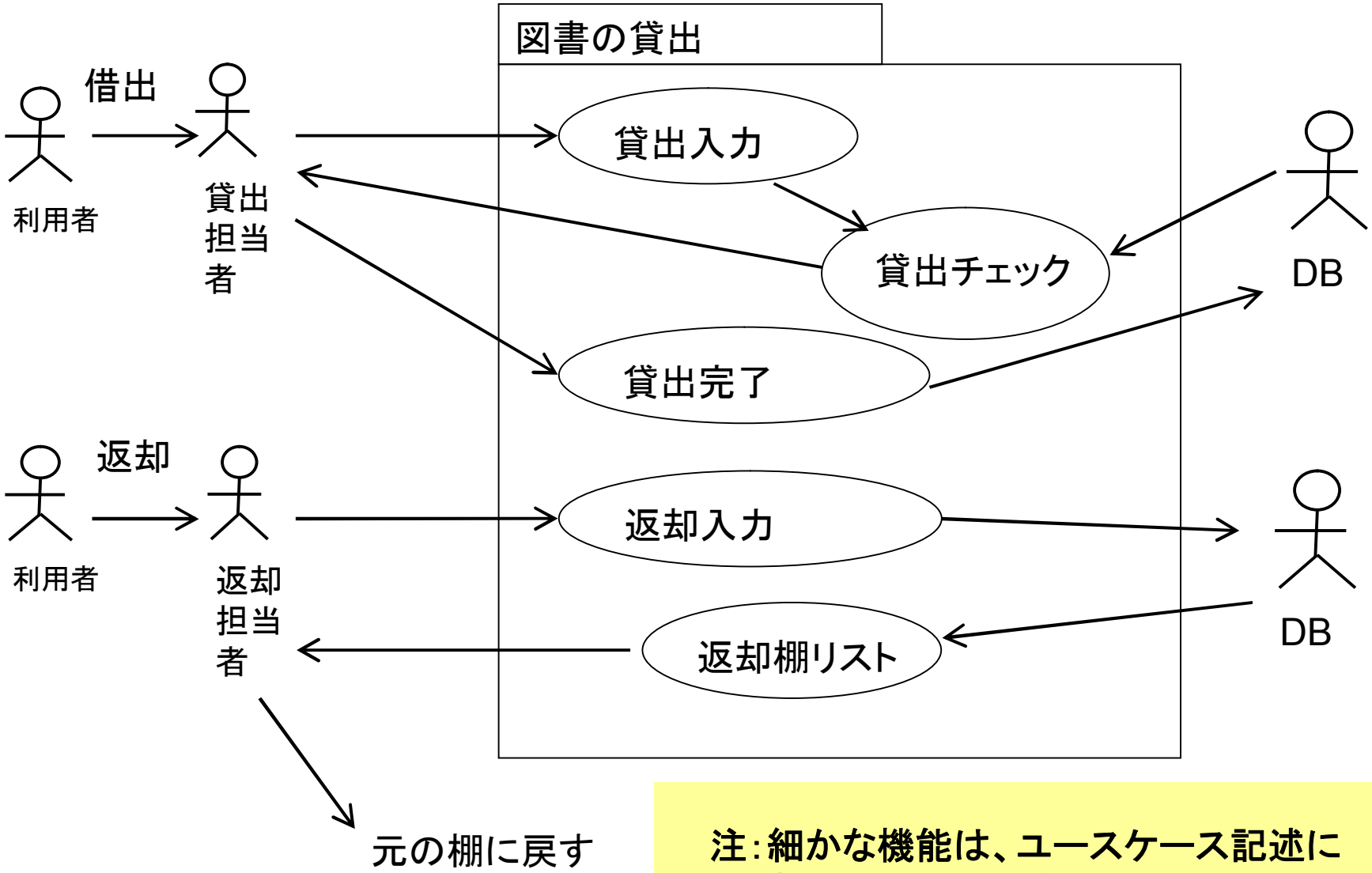
UMLによるモデリングの効果

- どのようなシステム仕様を持っているか表現可能。
- システム化対象を目に見える形で表し、
  開発者間でモデルの共有化ができる。
- システムの実装に関しての指示になる。
- 将来、自動プログラム製造の可能性がある。
- システムの仕様を標準化された文書で表し、
  運用・保守時に利用できる。

ユースケース図

・システム利用者の立場で機能（ユースケース）を取り出す。
・アクターの行為をよく分析する。
　ーシステムに対して、双方向で何をするのか
　　（入力、変更、通知、システムから知らせ・・・）
・アクターとして、外部のシステム、装置も取り出す。
・ユースケース作図の前に、アクターの役割を記述する。
・主たる利用者を優先して取り出す。
　　（システム管理者などは副アクターとして後で追加。）
・要件定義段階では、機能を細かく分けない。
　　（例）支払い機能を、現金／カード／小切手での支払い
　　などには分けない。
　　チケットの印刷、請求書のプリントなども分けない。

図書の貸出 （貸出、返却）ユースケース



注：細かな機能は、ユースケース記述に書くようにすると、簡潔で分かりやすい。

ユースケース記述

・各ユースケース毎に、目的、概要を記述する。

ユースケースの記述を詳細化することにより、例外事項、モレなどの発見が進む。

ユースケース記述書

ユースケース概要を記入xxxxxxx

目的　　：　xxxxxxxxxx

概要　　：　xxxxxxxxxx

前提条件　　：　xxxxxxxx

制約条件　　：　xxxxxxxx

相互対話の順序（シナリオ、処理手順）

1.　xxxxxxxxxx

2.　Xxxxxxxxxx

システム、アクターがやることを区別

全体

ユースケース名称

アクターの説明

ユースケースの説明

（何をするのか、条件は無いか、例外は無いか・・・）

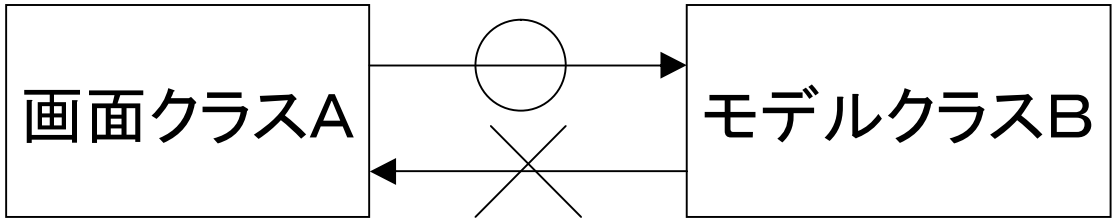
ユースケース毎に

クラスの取り出し方

- 基本は、要求仕様書、ユースケース記述書を元にして、名詞句に着目して、クラス候補を取り出す。
  （クラス、属性の元になるもの）
- クラス候補を取出したあとで、対象領域のなかで重要なモノからクラスとし、それと関係の深いモノを見出す。
- 意味が重なっているものは、名前を一つに統合する。
- ユーザーインターフェース記述書から、インターフェースクラスを取り出す。（例示、チケット予約画面、チケットプリンター・・・）
- 要件定義段階では、実装段階で必要となるクラスは未だ取り出さない。（個々のミドルウェアなど・・・）

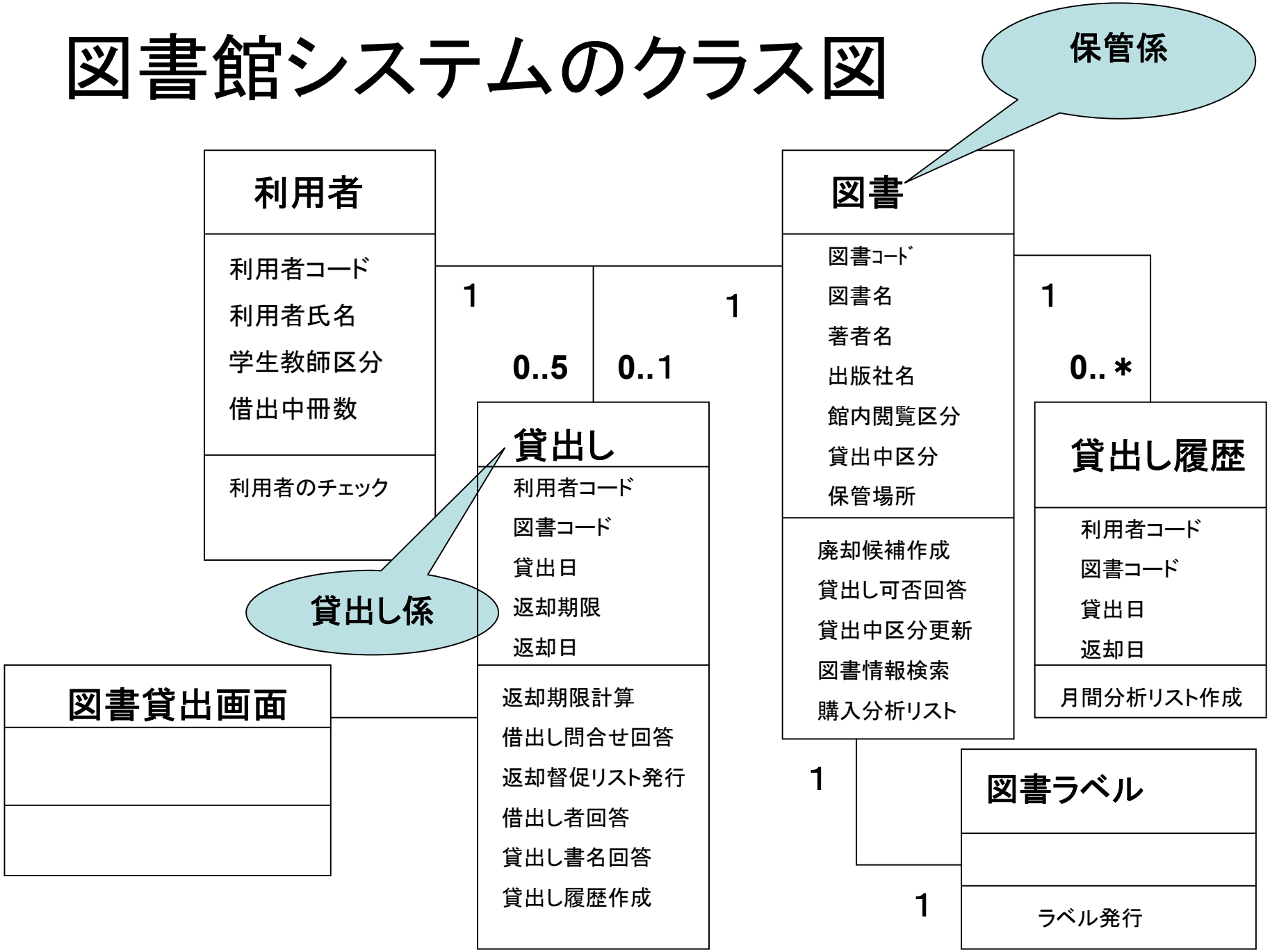
画面クラスの使い方

- モデルクラスとビュークラスを切り離す。
  ーモデルはシステムの内部の論理(データ、処理)
  ービューは、システムの見かけの部分(画面、帳票)
  従って、画面クラスは独立させる。
- 画面は変更が頻繁であり、モデルクラスから
  画面クラスへの関連はもたせないこと。



- 画面クラスには、イベント駆動の操作を準備する。
  ーどのボタンが押されたら、どの操作を実行する。
- 画面クラスの操作では、通常、ライブラリー中の
  部品を活用する。(スクロール、表示、ボタン、画面形状)

図書館システムのクラス図



画面設計書

チケット予約画面

日付 ↓

対戦相手 ↓

座席種別 ↓

枚数

クラス仕様書

システム名称

クラス名称

属性

属性名	型	属性の意味

操作

操作名	操作内容

よくあるクラス設定の間違い例

X　大きなクラス1つで済ませる （大きなプログラムを作成するの同じで、製造・保守が大変である）

X　機能に着目して、機能の数だけクラスを設ける

　　例示　　チケット予約、チケット変更、チケットキャンセル

X　属性をクラスに格上げする

　　例示　　車・鉛筆・洋服・靴などのサイズ、数値、色など

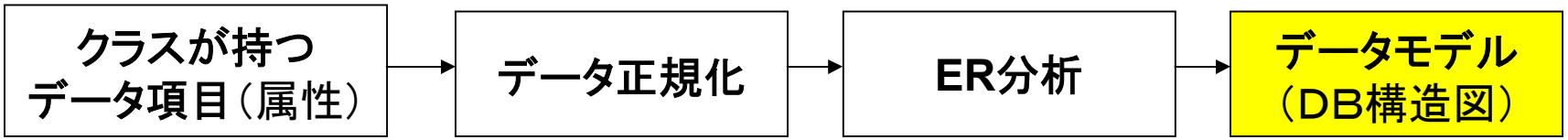
△　データベースをクラスとする

　　間違いでは無いが、内部設計の後半にクラスとして定義すればよい。

クラス図とデータモデル図

- 機械制御システム、OSシステムなど
  - ・データ量が少なく、クラス図のみで表現可能。
  - ・ただし、マルティメディア・システムでは、データ＋操作が一体の方が、即物的で扱い易い。（音楽、動画・・・）
- ビジネス・システム
  - ・データ使用頻度が高く、しかも量が多い。
  - ・ビジネス・システムにおいては、データ項目は永続性がある。
  - ・従って、クラスからデータ部分を独立させて、データモデルを作成する。

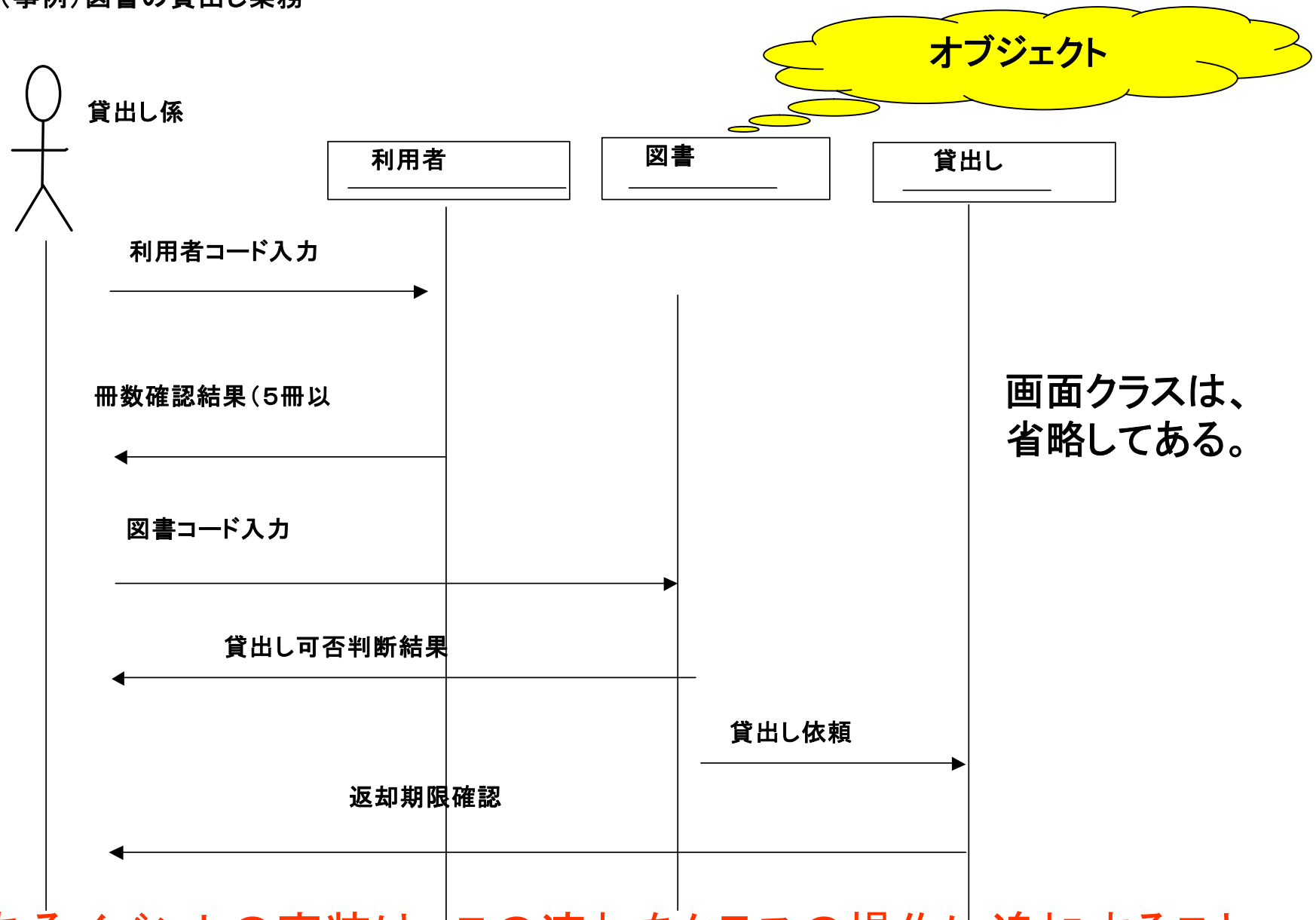
＜データモデルの作成手順＞



シーケンス図の作成

・基本は、ユースケース図、ユースケース記述書、
クラス図を元にした、メッセージの時系列表現である。
　－アクターを起点に、関係があるオブジェクト間の
　－イベントの流れを順番に見つけ出すことである。
・煩雑さを避けるために、イベント毎に、シーケンス図は
独立させて作成する方がよい。
・よくあるシーケンス図の間違い（役割分担）
　X　一つのクラスに役割が全て集中している。
　X　上からのメッセージに対して、別なことを返している。
　X　受けたメッセージをそのまま下流に流している。

（事例）図書の貸出し業務



画面クラスは、
省略してある。

- あるイベントの実装は、この流れをクラスの操作に追加することである。

基本は、機器の設計者、システムの設計者の立場で考える

ユースケース図

ー要件定義段階で作成する。

ー要求を分析し、ユーザーが何を要求しているのかを書き出す。

ーその要求を実現するためには、どんな機能が必要かを取り出して書き出す。

ーその機能をユースケース図、ユースケース記述、インターフェース記述にまとめる。

・制御機器の場合は、その機器の機能（働き）を取出す。

・ビジネスシステムの場合は、〇〇係／〇〇担当者の仕事内容。

・ クラス図

ー外部設計段階で作成する。

ー機器またはシステムの内部構造のこと。

ー機器の場合は、機器の主要な構成部品である。

ービジネスシステムの場合は、通常〇〇係／〇〇担当者がクラスに当たる。

・ 〇〇係／〇〇担当者の代わりにやるのはプログラムである。

ーその機器の構成部品の役割が操作であり、〇〇係／〇〇担当者の頭の中にあるデータとかノートが属性であり、仕事の内容が操作である。

ーそれをクラス図、クラス仕様書のまとめる。

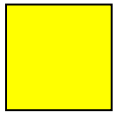
シーケンス図

ー外部設計で作成する。

ー機器の場合は、構成部品間の信号の流れ。

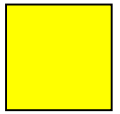
ービジネスシステムでは、クラス間の依頼書／メッセージの流れをまとめたもの。

４つの図表の役目



- ユースケース図
  - ・具体的なシステム機能を表現
    （受入→受入指示書、検収報告書）
- クラス図
  - ・上記の機能を実現するためのシステム構造
- データモデル図
  - ・データベースの構造
- シーケンス図
  - ・クラスが持つシステム機能の使用順序

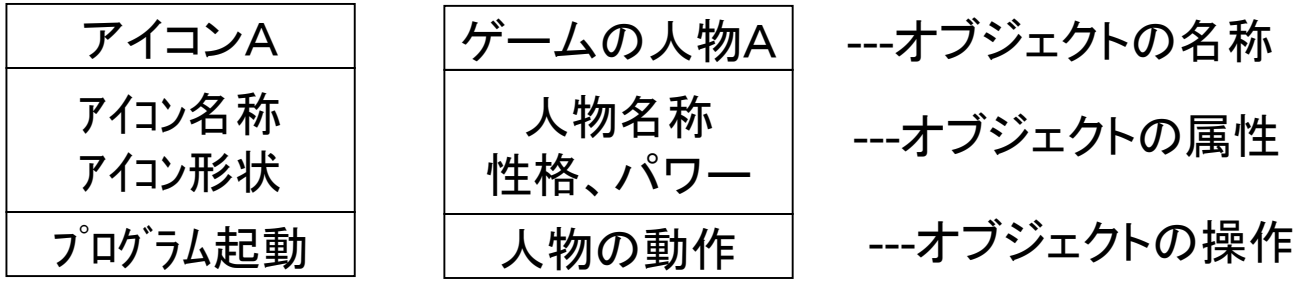
システム化の工程と技法



- 上流工程（要件定義、外部設計）
  - ・オブジェクト指向的な考えで、UMLを使用
  - ・ビジネスシステムでは、データベース設計が必須
    （クラスから属性を取り出しDB化する）
- 下流工程（特に内部設計）
  - ・開発効率、保守のし易さ、開発規模を少なく
  - ・プログラムの部品化、再利用をすすめる
  - ・差分プログラミングをすすめる
    （モジュール化、フレームワーク化（ベースソフト））

オブジェクト指向の適用可否

- 適用が向いているもの
  - ・機器制御、ゲームソフト



- そのままでは適用が向いていないもの
  - ・ビジネスシステム用の情報システム

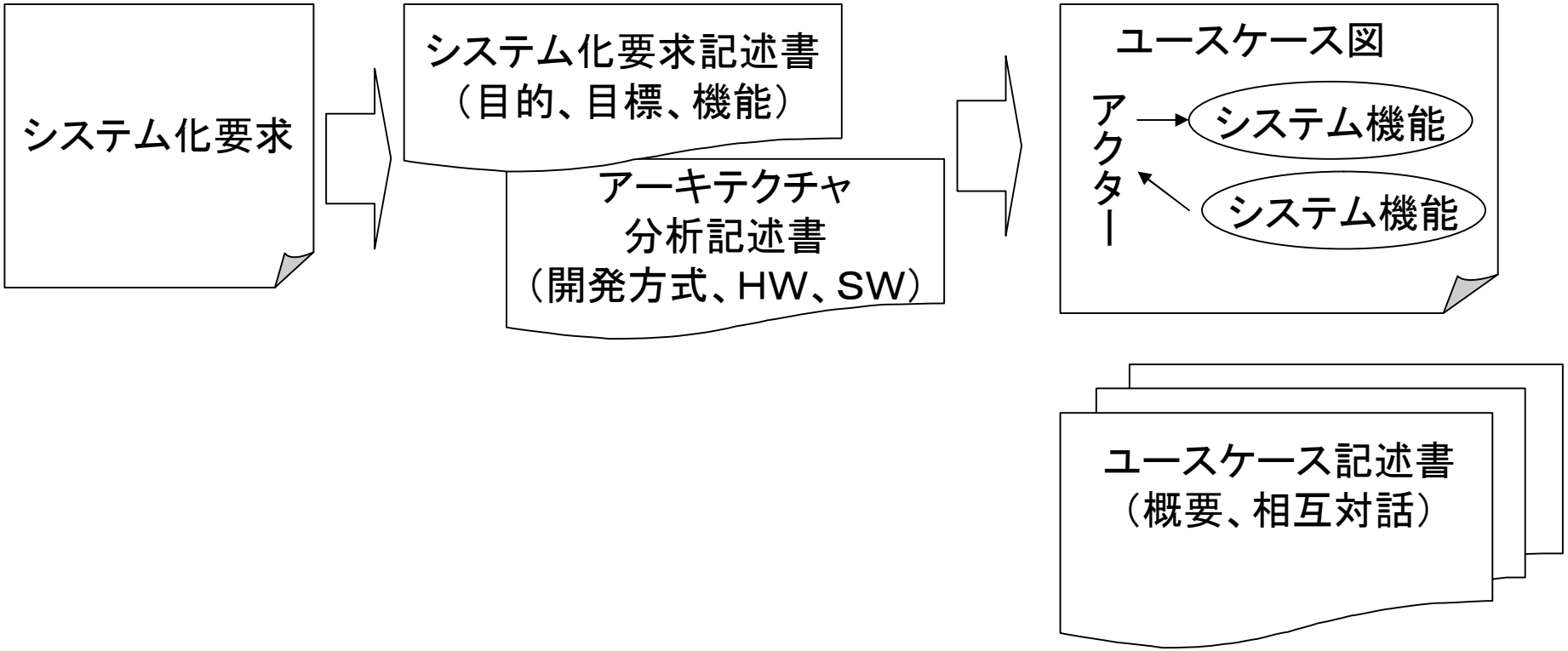
    ただし、属性を除き、操作部分については、
    システム機能の細分化により、部品化や
    差分プログラミングは可能

    （業務ロジック、DB操作、画面制御、データ検証、
    認証機能、フォーマット変換、Excel出力・・・）

要件定義のプロセス

・ユーザーのシステム要求を分析して、システム化要求をまとめる。

・それをもとにして、システム機能をユースケース図、ユースケース記述書にまとめる。

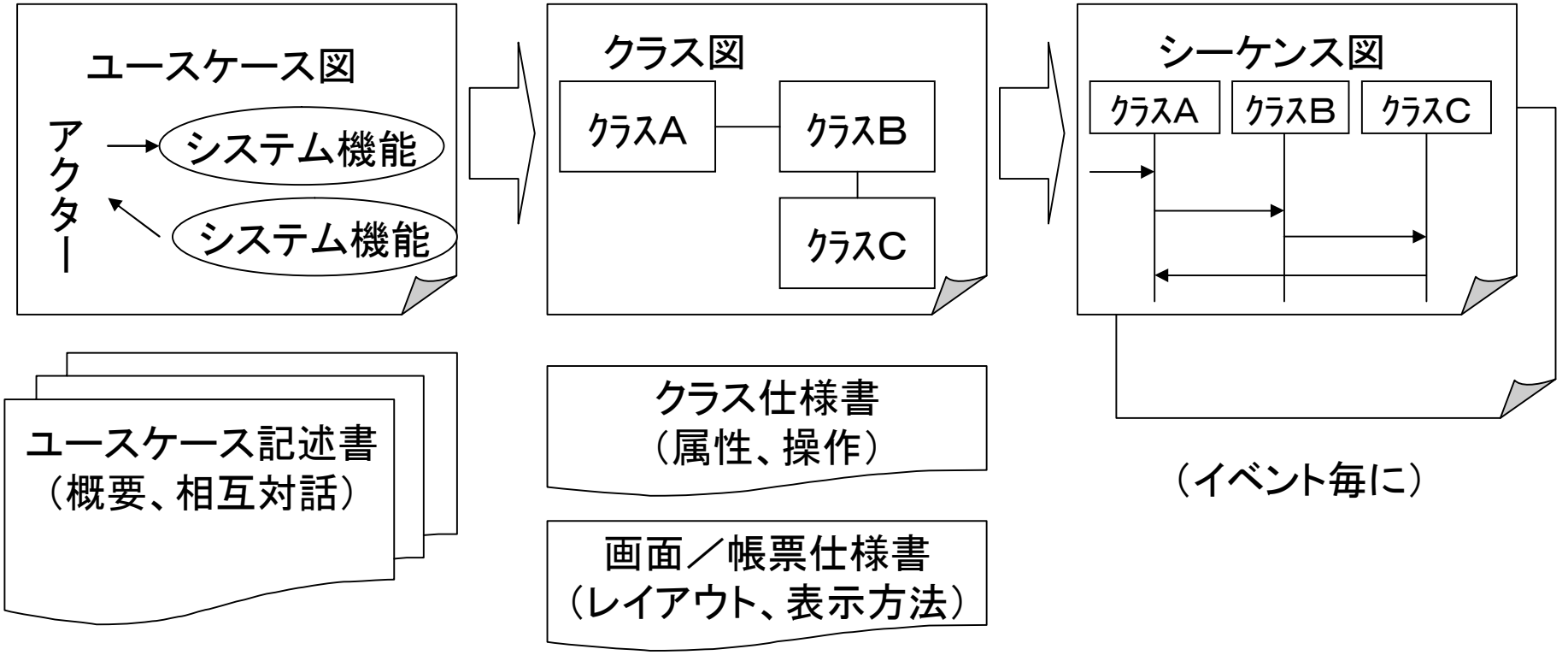


外部設計のプロセス

・要件定義で作成したシステム機能定義をもとにして、オブジェクトモデルを作成する。

（静的モデル設計（クラス図）、クラス仕様書、画面／帳票仕様書、

動的モデル設計（シーケンス図））



内部設計のプロセス

・外部設計で作成したオブジェクト・モデルをシステム実装レベルまで具体化する。

（クラスの詳細化　＝　特化（下位クラス作成）、汎化（共通クラス作成））

（クラスの分解　　＝　インターフェースクラスの追加）

（画面クラスの詳細化　＝　画面利用時のモレを無くす）

（クラスの属性をもとにして正規化・ERモデル化によりデータベース構造）

（属性の追加・修正）

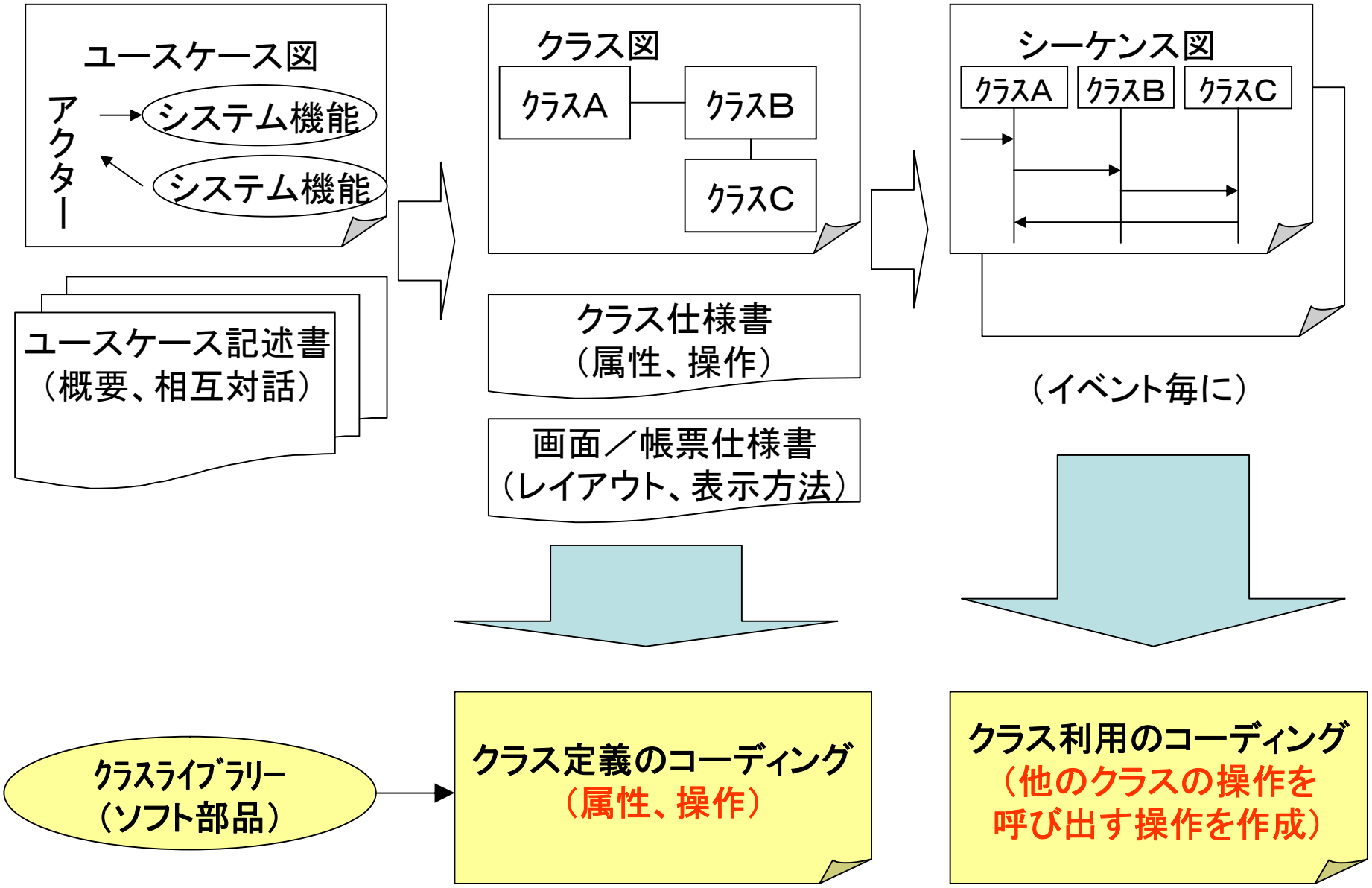
（操作の追加・修正）

・シーケンス図の詳細化→→→→→アプリケーションのプログラム設計になる。

（上記で変更したクラスを、シーケンス図上でも変更する）

（上記で変更した操作を、クラスが発するメッセージに反映する。）

設計とプログラミングの関係

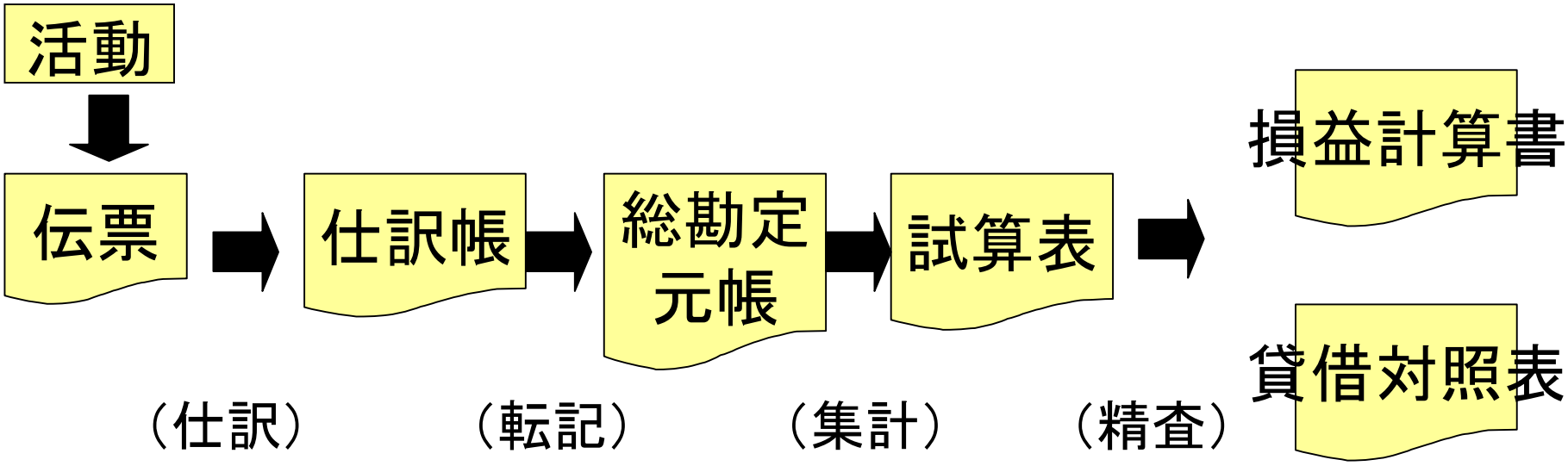


経理業務の分類

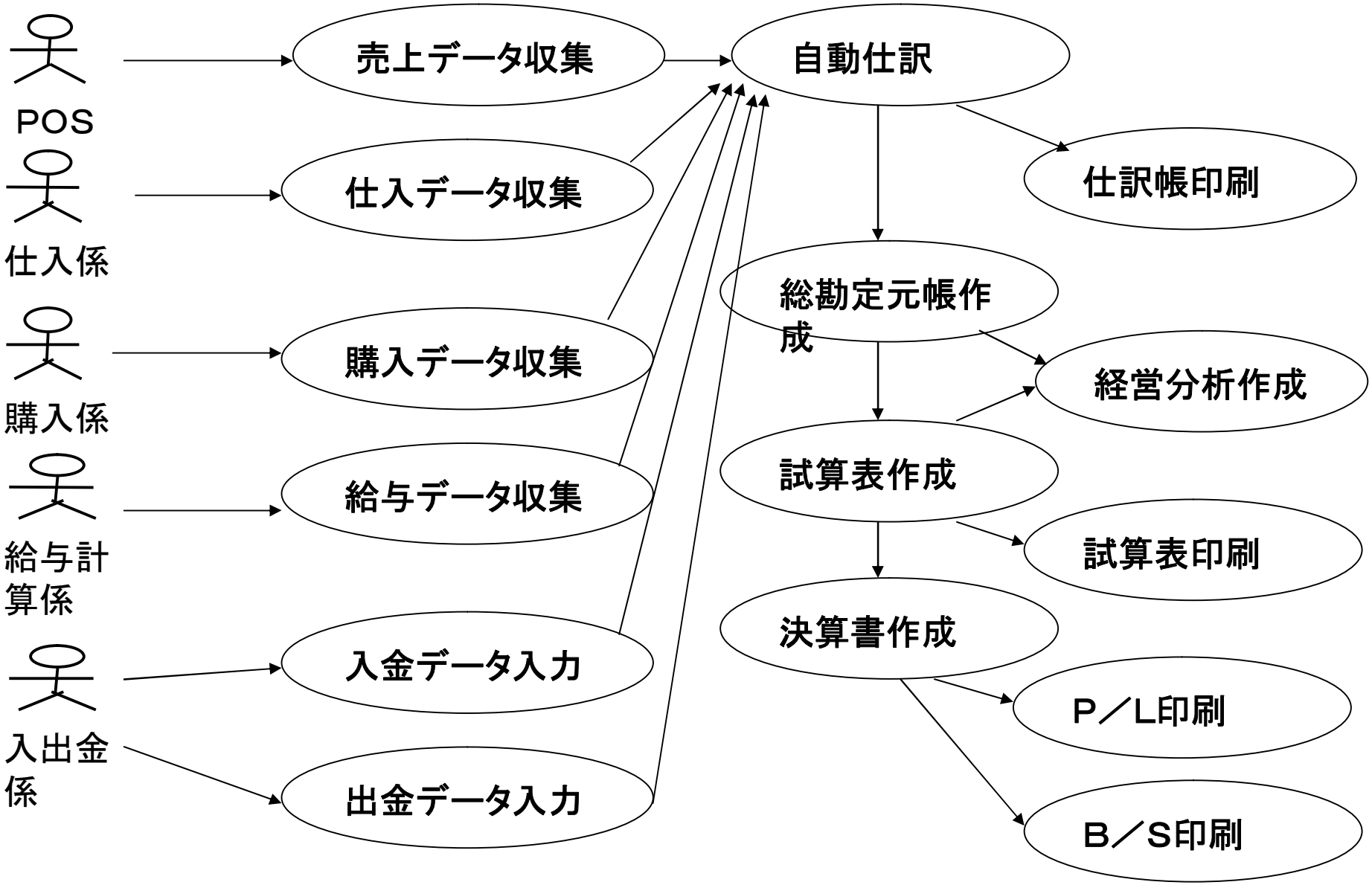
分類	目的	内容
財務会計 （外部報告会計） （簿記会計）	・外部への報告 ・税金の計算	・伝票もとに簿記（仕訳、総勘定元帳、残高表） ・原価計算 ・決算書作成（P/L、B/Sなど）
資金会計	・資金の動きの把握 ・資金の有効活用	・出納業務、未収金の回収 ・資金計画、資金繰り、資金調達 ・資産管理、棚卸
管理会計 （内部報告会計） （業務改善会計）	・経営支援 ・部門運営支援 ・業務改善	・経営計画、利益計画、コスト管理 ・予算管理、部門予算管理 ・問題点把握、改善点把握

簿記業務の流れ

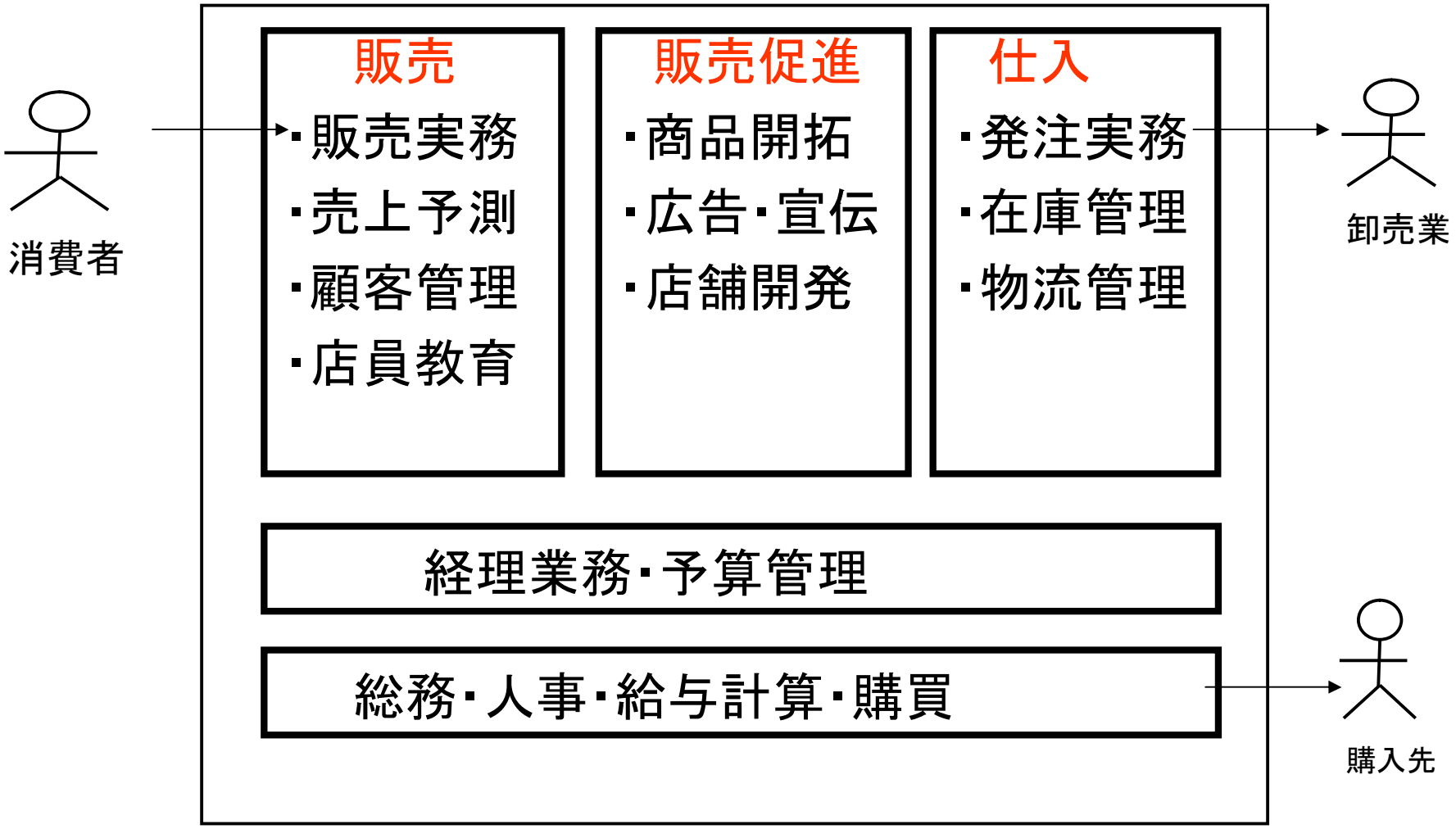
各活動結果の伝票を元にして、所定の規則に則って、資産の活用状況を明らかにする。



経理業務のユースケース(事例)



小売業の業務体系


消費者
販売
・販売実務
・売上予測
・顧客管理
・店員教育
販売促進
・商品開拓
・広告・宣伝
・店舗開発
仕入
・発注実務
・在庫管理
・物流管理
卸売業
経理業務・予算管理
総務・人事・給与計算・購買
購入先

販売業務のクラス図の基本

・データ正規化の観点からみれば、容易にクラスを導出できる。

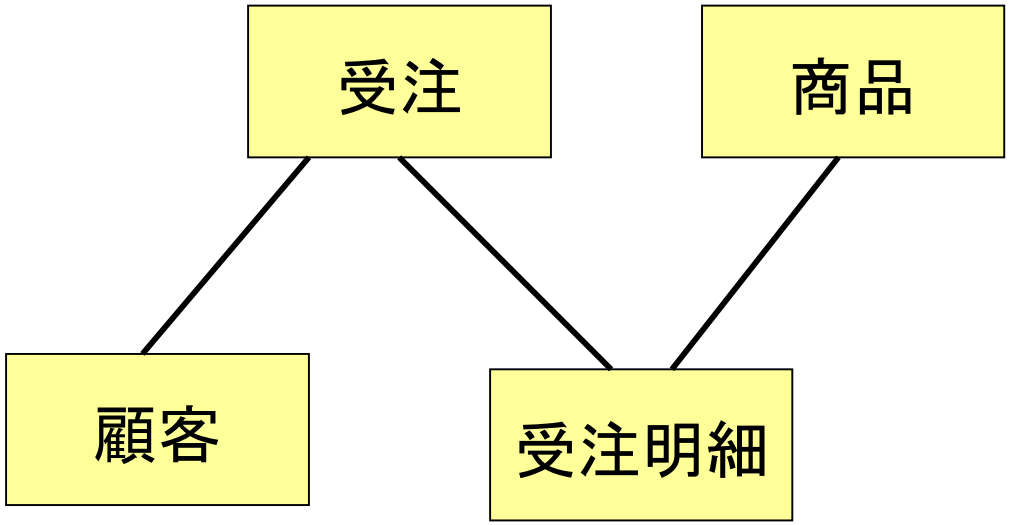
受注（注文、売上、レシート）

顧客名	受注No	受注日

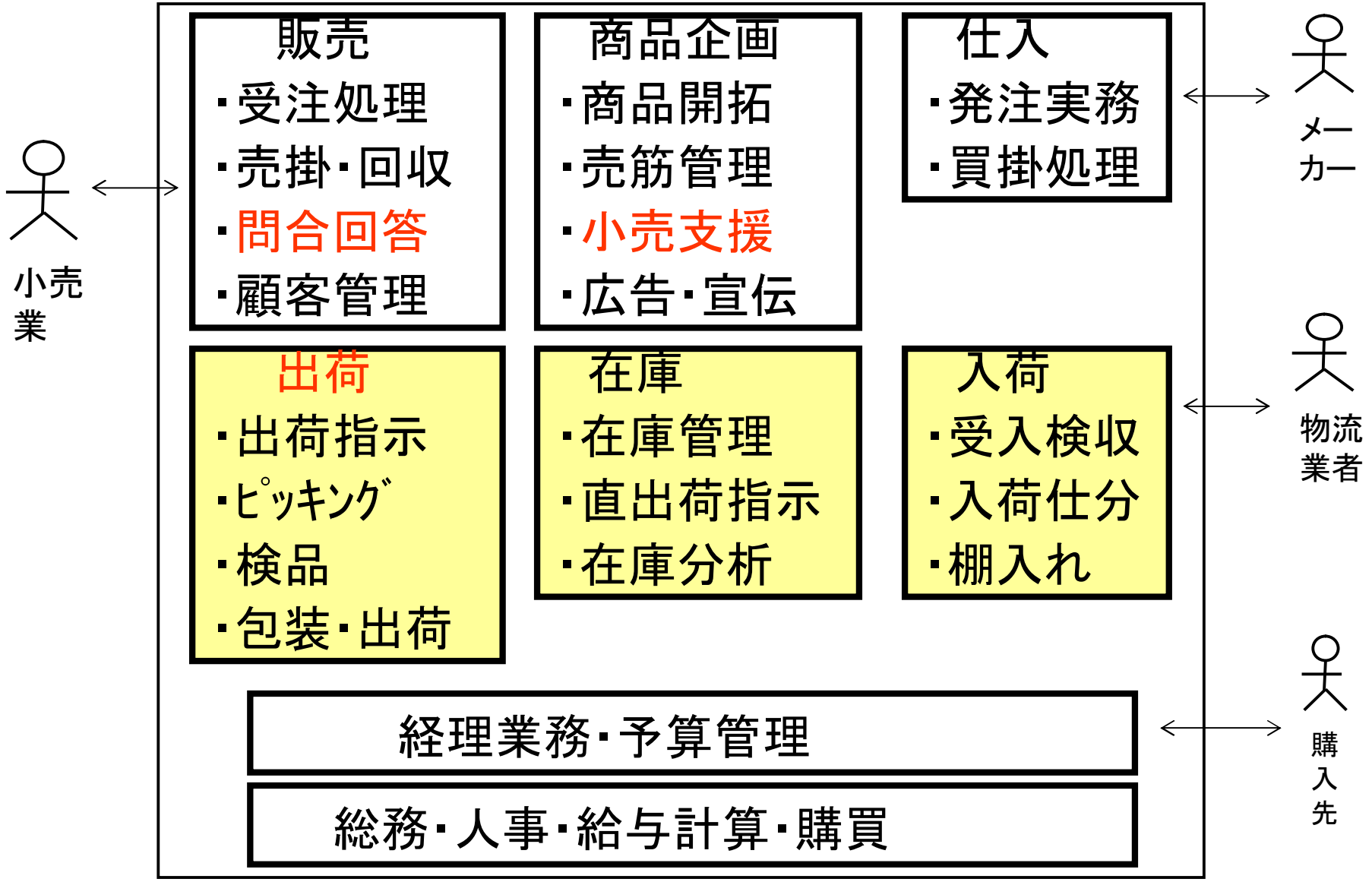
商品	単価	数量	金額

合計金額

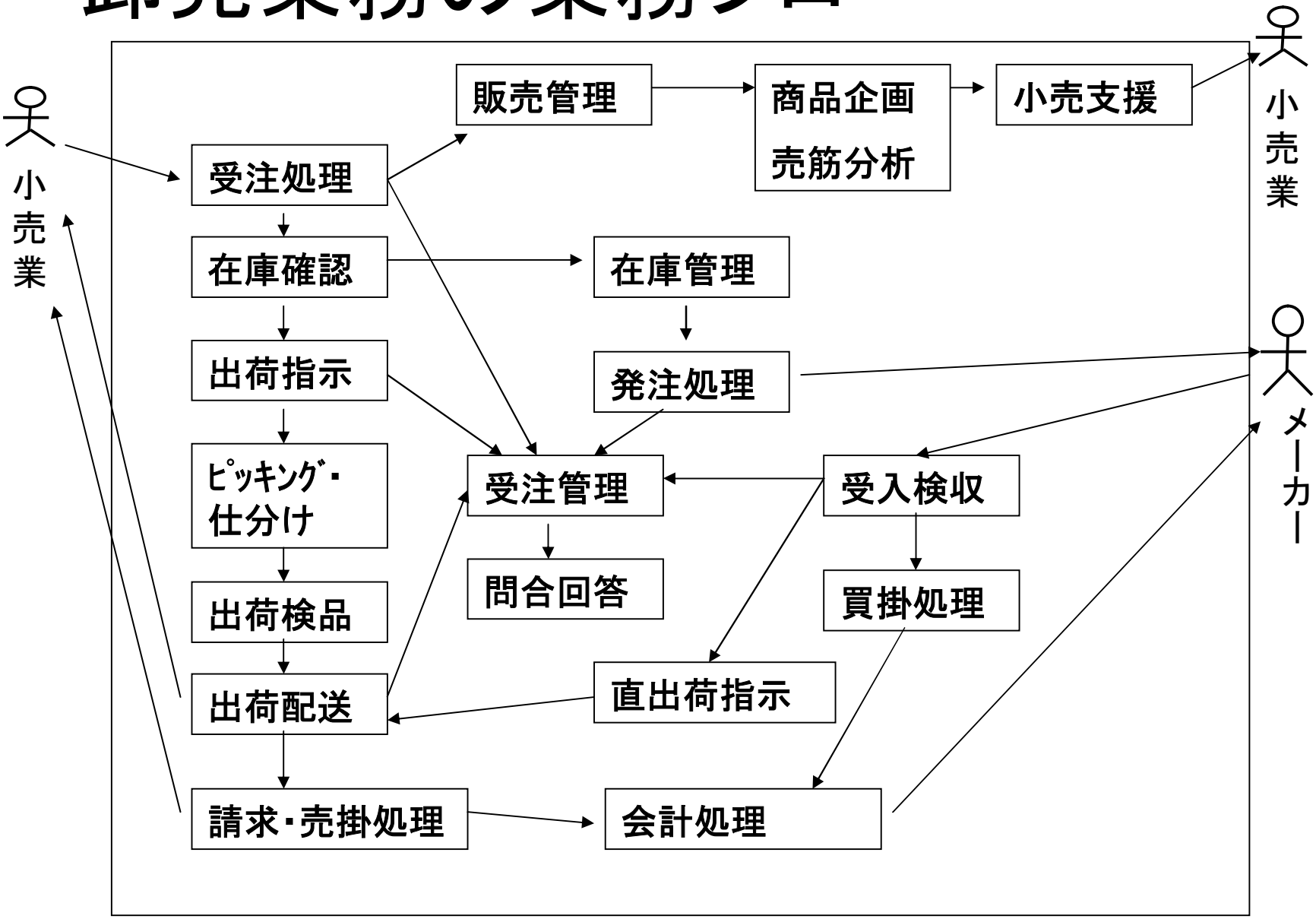
データの正規化、クラス図



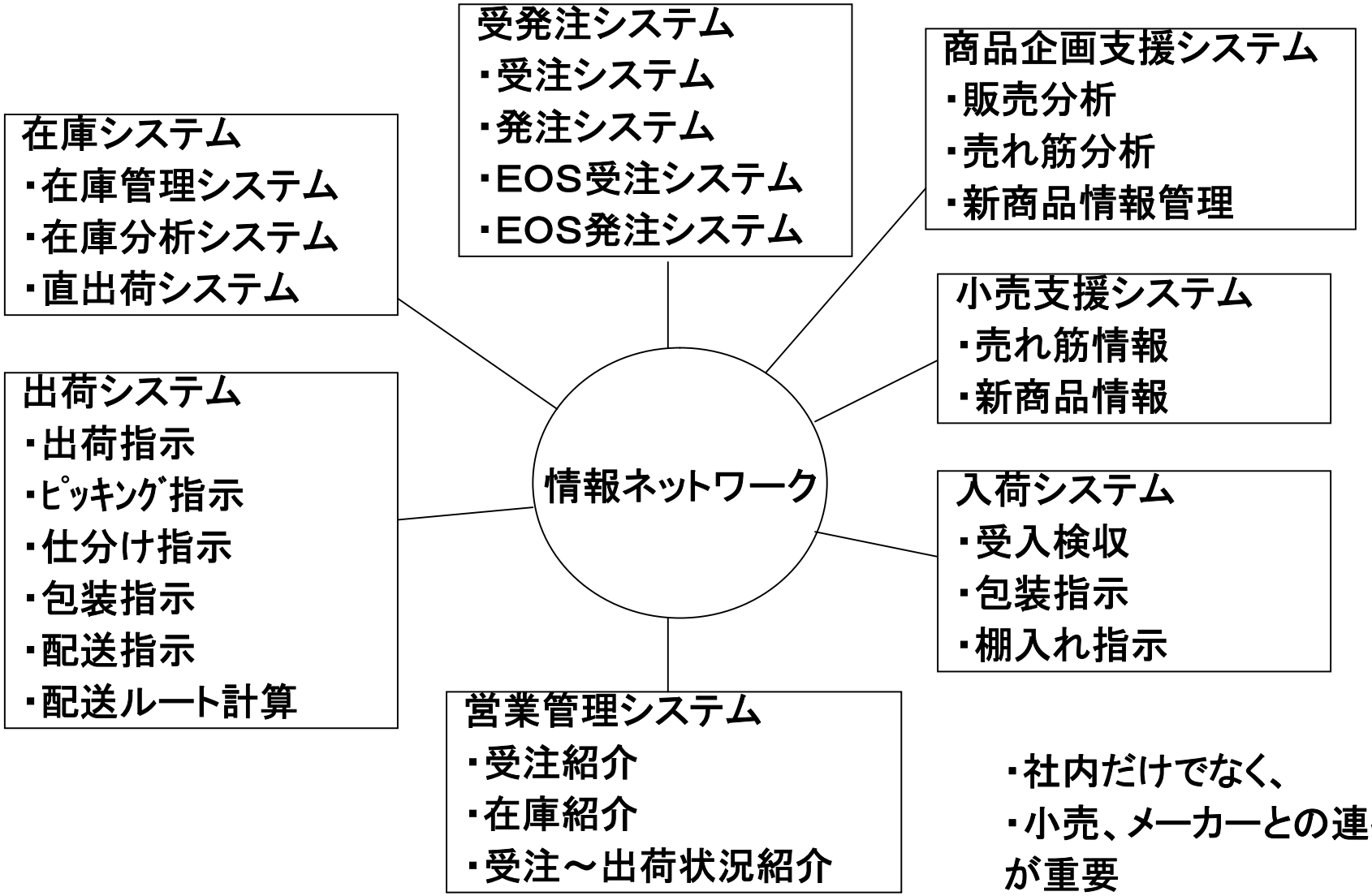
卸売業の基本業務体系


販売
・受注処理
・売掛・回収
・問合回答
・顧客管理
商品企画
・商品開拓
・売筋管理
・小売支援
・広告・宣伝
仕入
・発注実務
・買掛処理
小売業
メーカー
出荷
・出荷指示
・ﾋﾟｯｷﾝｸﾞ
・検品
・包装・出荷
在庫
・在庫管理
・直出荷指示
・在庫分析
入荷
・受入検収
・入荷仕分
・棚入れ
物流業者
経理業務・予算管理
総務・人事・給与計算・購買
購入先

卸売業務の業務フロー

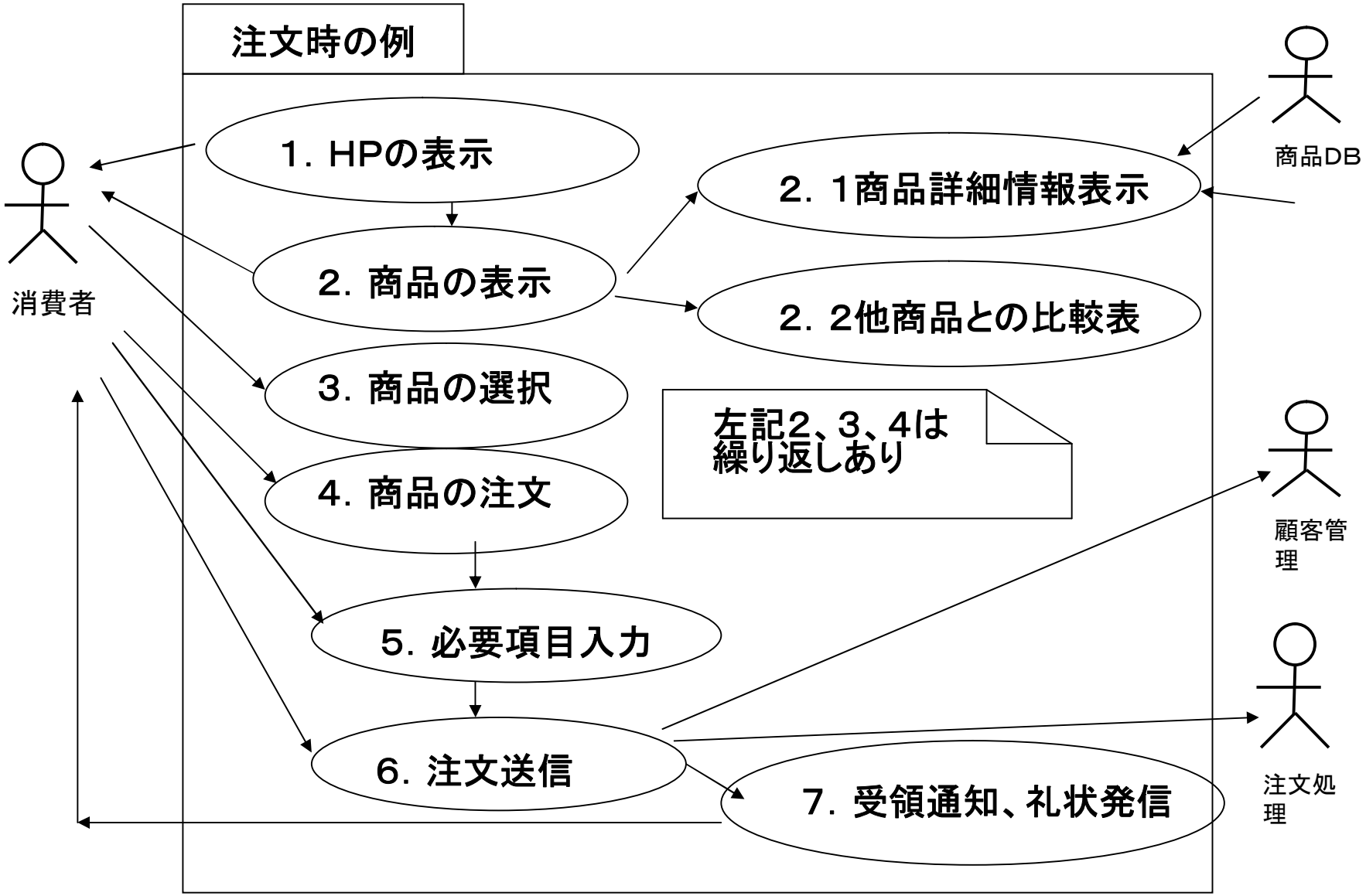

小売業
販売管理
商品企画
売筋分析
小売支援
小売業
受注処理
在庫確認
在庫管理
出荷指示
発注処理
メーカー
ﾋﾟｯｷﾝｸﾞ・
仕分け
受注管理
受入検収
問合回答
買掛処理
出荷検品
出荷配送
直出荷指示
請求・売掛処理
会計処理

卸売業の統合システム

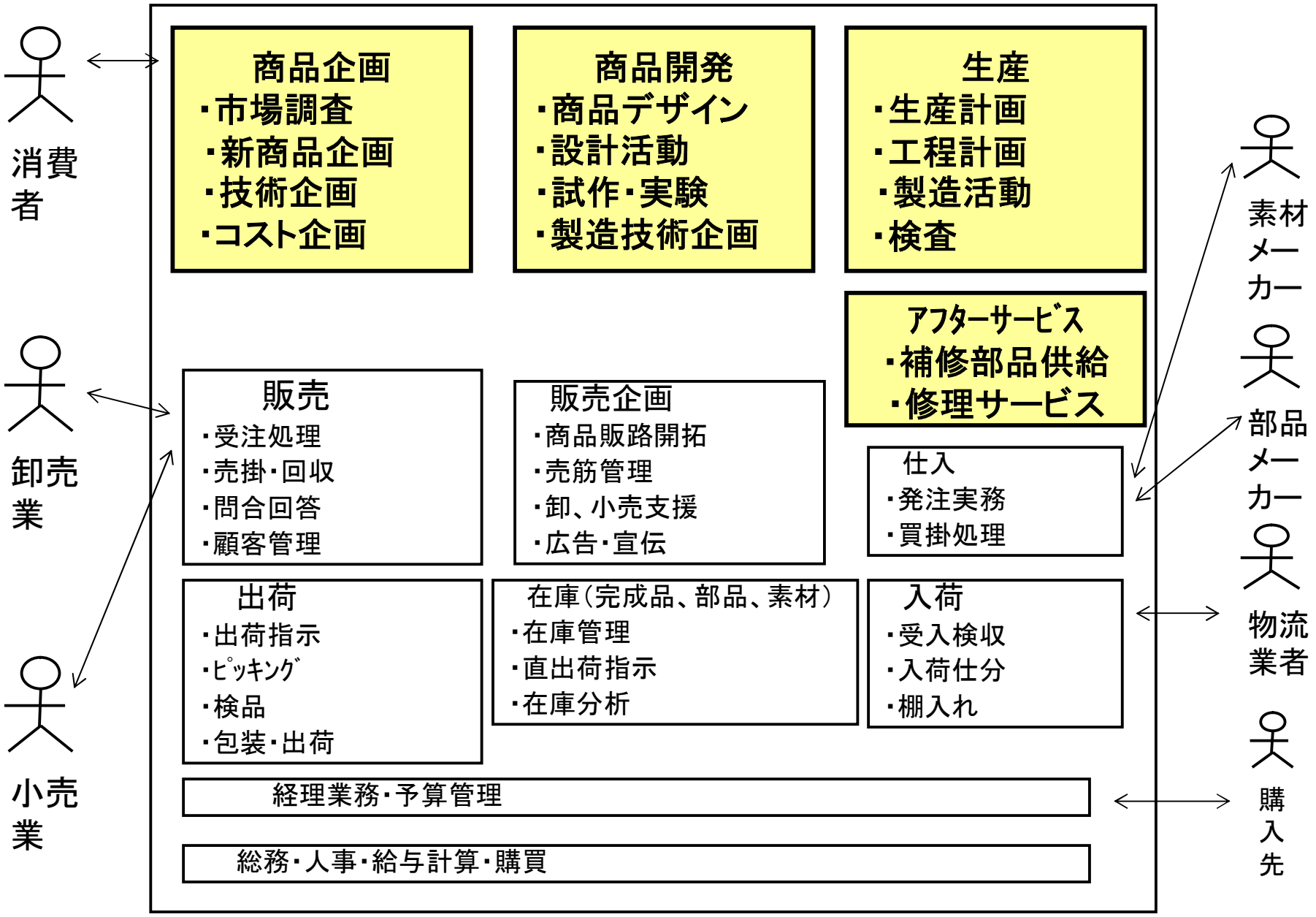


・社内だけでなく、
・小売、メーカーとの連携が重要

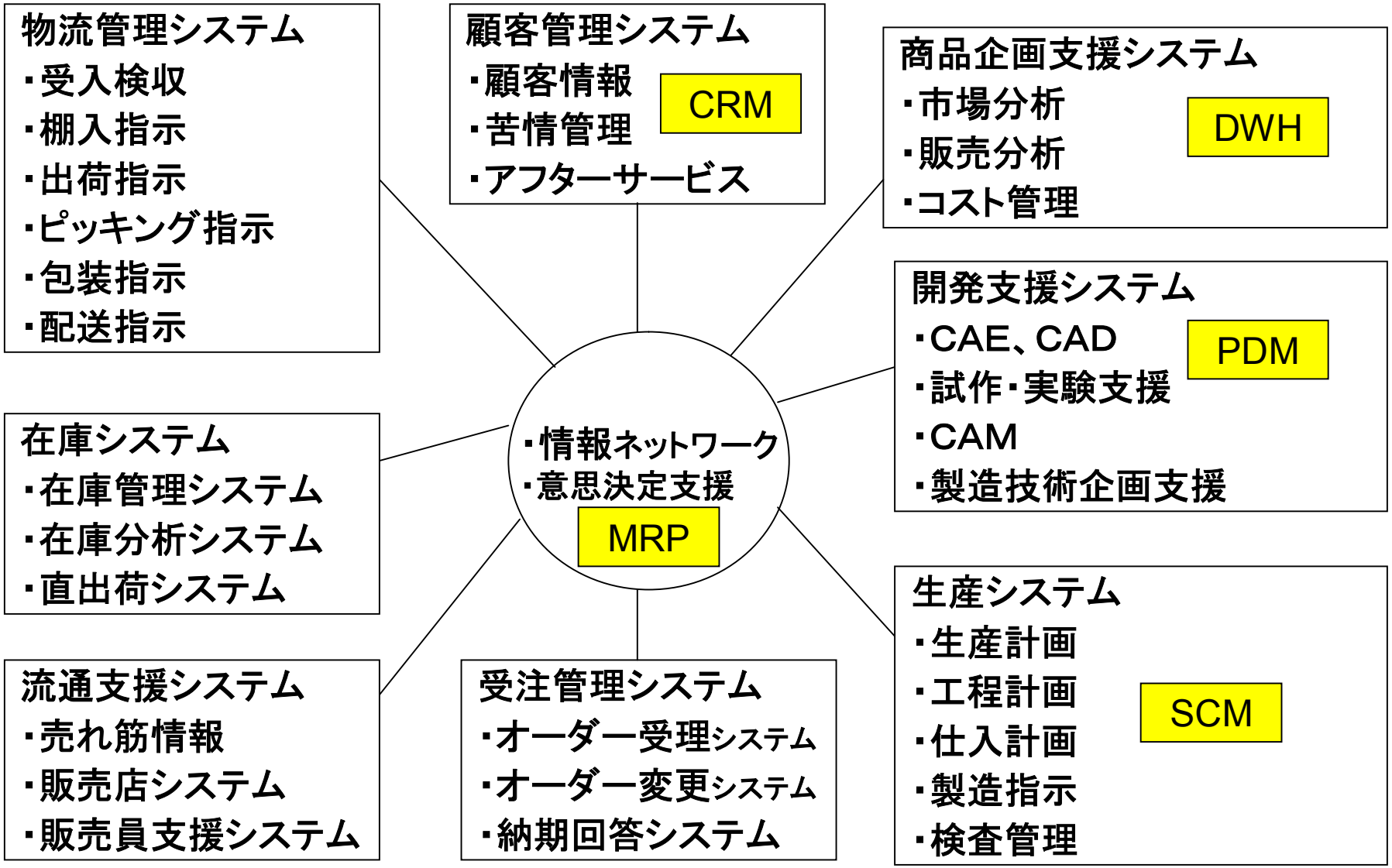
インターネット店舗システムのユースケース



製造業の業務体系


消費者
卸売業
小売業
商品企画
・市場調査
・新商品企画
・技術企画
・コスト企画
商品開発
・商品デザイン
・設計活動
・試作・実験
・製造技術企画
生産
・生産計画
・工程計画
・製造活動
・検査
ｱﾌﾀｰｻｰﾋﾞｽ
・補修部品供給
・修理サービス
販売
・受注処理
・売掛・回収
・問合回答
・顧客管理
販売企画
・商品販路開拓
・売筋管理
・卸、小売支援
・広告・宣伝
仕入
・発注実務
・買掛処理
出荷
・出荷指示
・ﾋﾟｯｷﾝｸﾞ
・検品
・包装・出荷
在庫（完成品、部品、素材）
・在庫管理
・直出荷指示
・在庫分析
入荷
・受入検収
・入荷仕分
・棚入れ
経理業務・予算管理
総務・人事・給与計算・購買
素材メーカー
部品メーカー
物流業者
購入先

製造業の統合システム


物流管理システム
・受入検収
・棚入指示
・出荷指示
・ピッキング指示
・包装指示
・配送指示
顧客管理システム
・顧客情報
・苦情管理
・アフターサービス
CRM
商品企画支援システム
・市場分析
・販売分析
・コスト管理
DWH
開発支援システム
・CAE、CAD
・試作・実験支援
・CAM
・製造技術企画支援
PDM
・情報ネットワーク
・意思決定支援
MRP
在庫システム
・在庫管理システム
・在庫分析システム
・直出荷システム
生産システム
・生産計画
・工程計画
・仕入計画
・製造指示
・検査管理
SCM
流通支援システム
・売れ筋情報
・販売店システム
・販売員支援システム
受注管理システム
・オーダー受理システム
・オーダー変更システム
・納期回答システム

アイデアの出し方

－アイデア発想法

・つめ込む（ネタ、ルート、媒体）

・絞り込む（重点、重要）

・体系化　（関連付け、構造化）

・追加する（話す、人から）

－5W3Hで発想する

・What、Why、When、Who，Where

・How to

・How much

・How many

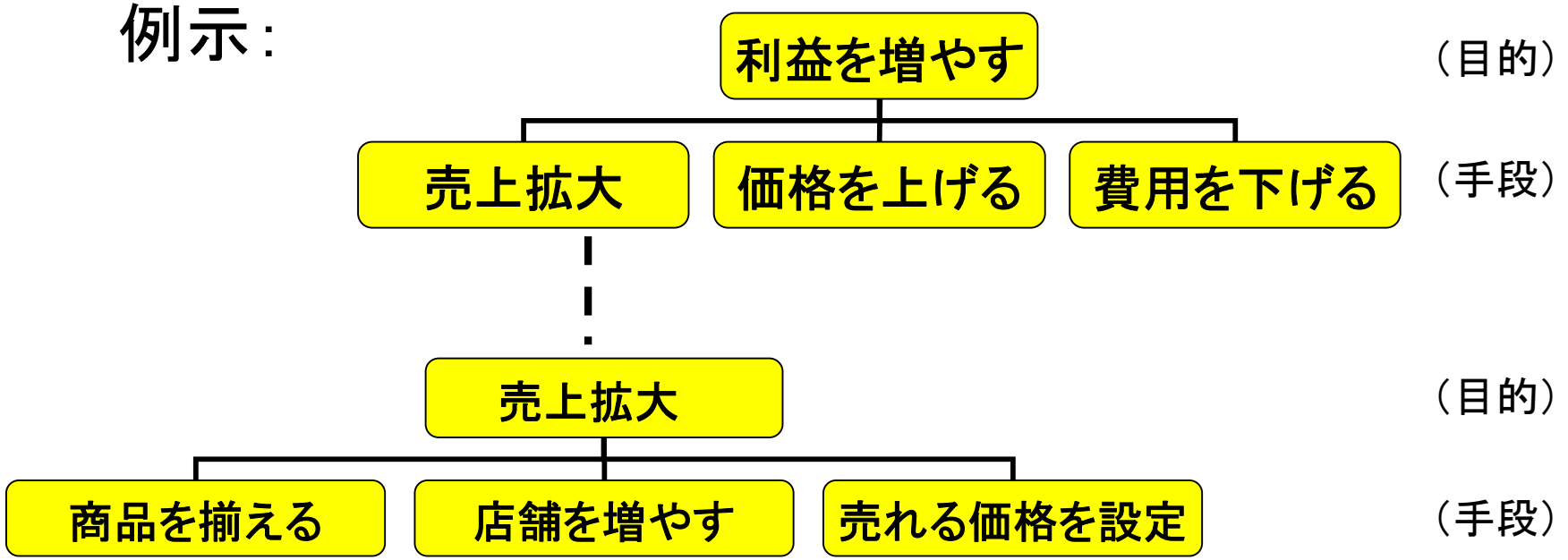
目的発想法----（村上　哲大氏）

・目的を決めて、その達成手段を考える。（機能分析）

・問題点を見つけて、その原因を探る。（因果分析）

（カードに書く、WHYを繰り返す、原点に帰る）

例示：



目的発想法の下方展開の方法

－機能に着目（つまり役割に着目）

・XXXが必要、正確さ、迅速さ、安く

－業務プロセスに着目（順序、手順）

－資源に着目（人、モノ、金、情報、時間）

－管理サイクルに着目（Plan-Do-See）

－現場に着目（数量、品質、コスト、納期・・・）

－内容の説明型（下位で上位の説明をする）

リテラシ（素養）

従来のリテラシ（素養）
・読み、書き、そろばん、話す、教養

情報技術リテラシ（ITリテラシ）
＜狭義＞コンピュータを操作できる
＜広義＞ ・情報ネットワークを活用して、
・必要な情報を収集・整理・加工・分析し、
・本質をつかんで情報を発信できる能力。
・ITを技術として管理、評価できる能力。

仕事のリテラシ
・一時には、一つのことに集中する
・立場を変えて考えてみる力（相手、お客）
・人との協調、チームメンバーシップ
・考えを紙に書いて整理する
・問題解決能力（問題点→原因究明→解決策）
・目標達成能力（目的→手段の連鎖発想）

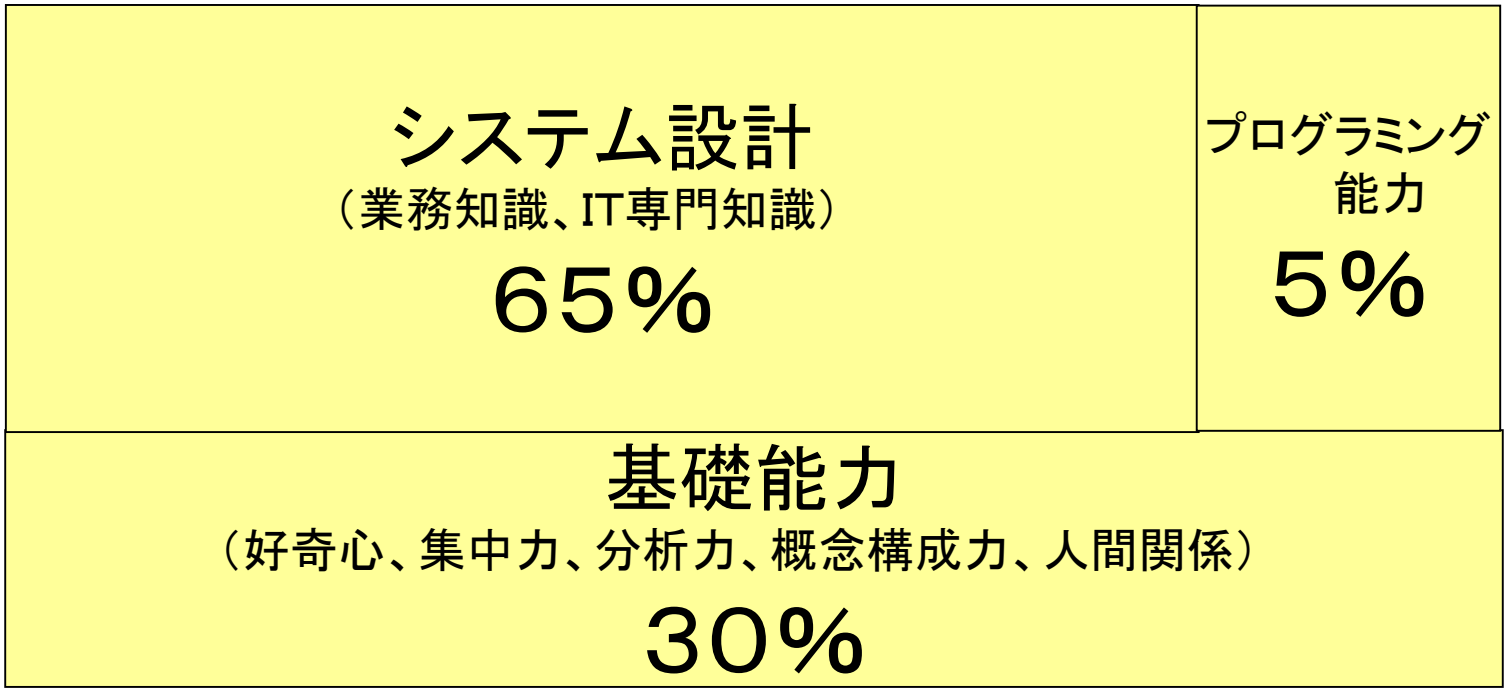
ベースのリテラシ
・自分の考えを持つ（本を読む＝考える）
・自分は、何がしたいかを考える力
・文章表現の能力（構想力、構成、文書化）
・うそを見抜く力
・他人の権利を侵害しない態度
・技術を評価して、何に使うべきかを考える力

必要なITテクノロジー （例示）

・B2C（Business to Customer）
　　－ｾｷｭﾘﾃｨ（Fire Wall、VPN、SSL、電子証明書・・・）
　　－ﾕｰｻﾞｰｲﾝﾀｰﾌｪｰｽ（i-mode、ﾌﾞﾗｳｻﾞｰ・・・）
　　－ｻｰﾊﾞｰ(Webｻｰﾊﾞｰ、ｱﾌﾟﾘｹｰｼｮﾝｻｰﾊﾞｰ、
　　　　DBMS、Java、JavaBeans、OS・・・)
・B2B（ Business to Business ）
　　－ｾｷｭﾘﾃｨ（VPN、SSL、電子証明書・・・）
　　－ﾃﾞｰﾀﾚﾍﾞﾙの連携（XML、XSL、XSQL・・・）
　　－インターフェース（FTP、HTTPS、SMTP、メッセージ・
　　キューイング・・・）
・コラボレーション
　　－ワーフローサーバー、ネットミーティング、メールサーバー
　　－データ共有（DBMS）
・その他要素
　　－OS（UNIX、Windows－NT、LINUX）
　　－運営（リモート障害監視、ｻｰﾊﾞｰ集中監視・・・）

SEの能力

- アプリケーションSEは、上流のシステム設計能力が主
- プログラミング能力は無くてもSEが務まる



SEの心構え

- 仕事の進め方
  - ー何事も計画を立ててから進める
  - ー書き出してみる（リスト、関連図）
  - ー自分の考えを話し、人からアイデアをもらう
  - ー5W3H（What, Why, Who、When、Where, How to, How much, How many）
- 筆者が目指して来たシステム
  - ー実務レベル（システムが指示した通りにやれば業務がうまく回る）
  - ー管理レベル（うまく回っていない業務は、その事実・原因がわかり有効な手が打てるような情報サービス）
  - ー企画レベル（製造・販売現場の生の情報をタイムリーに提供する）
- SEの心構え
  - ー好奇心→もっと知る→対象業務を好きになる
  - ー忍耐力、持久力、スタートしなければゴールに着かない
  - ープラス思考（入って来る全ての情報を「快」と捉えるクセ）
- トレンド
  - ーデータベースのスキルを持つエンジニア
  - ー業務系システムの構築・運用経験者
  - ープロジェクト・リーダー経験者など

“力のある”エンジニアへの需要の伸びは顕著