情報システム開発の手法2
（オブジェクト指向のアプローチ）

(情報システム開発論、第４回講義)

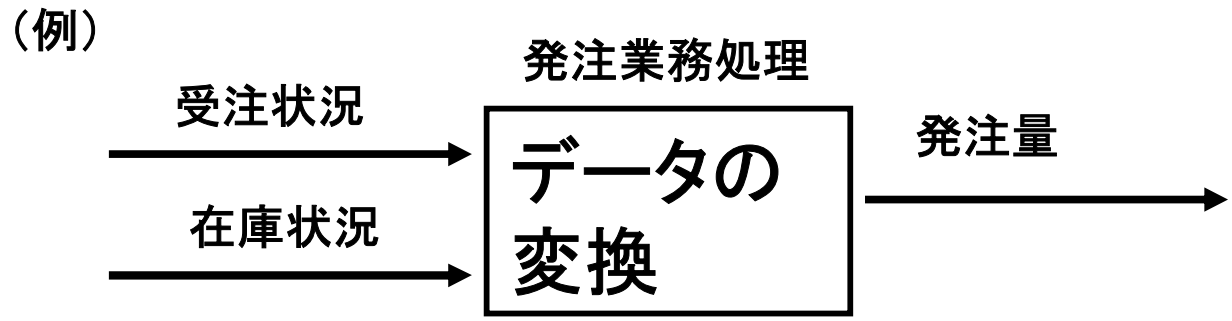
URL　http://homepage3.nifty.com/suetsuguf/

Email　fwhy6454@mb.infoweb.ne.jp

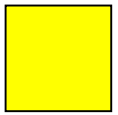
作成者　　末次　文雄　©

復習：DOA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。（処理は変わるが）
- 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。



復習：DOA開発手法



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
- 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）

②データの正規化

③ER図（Entity-Relationship Diagram）

目次（情報システム開発手法２）

1．オブジェクト指向とは

1．1　オブジェクト指向の特徴

①発想の原点

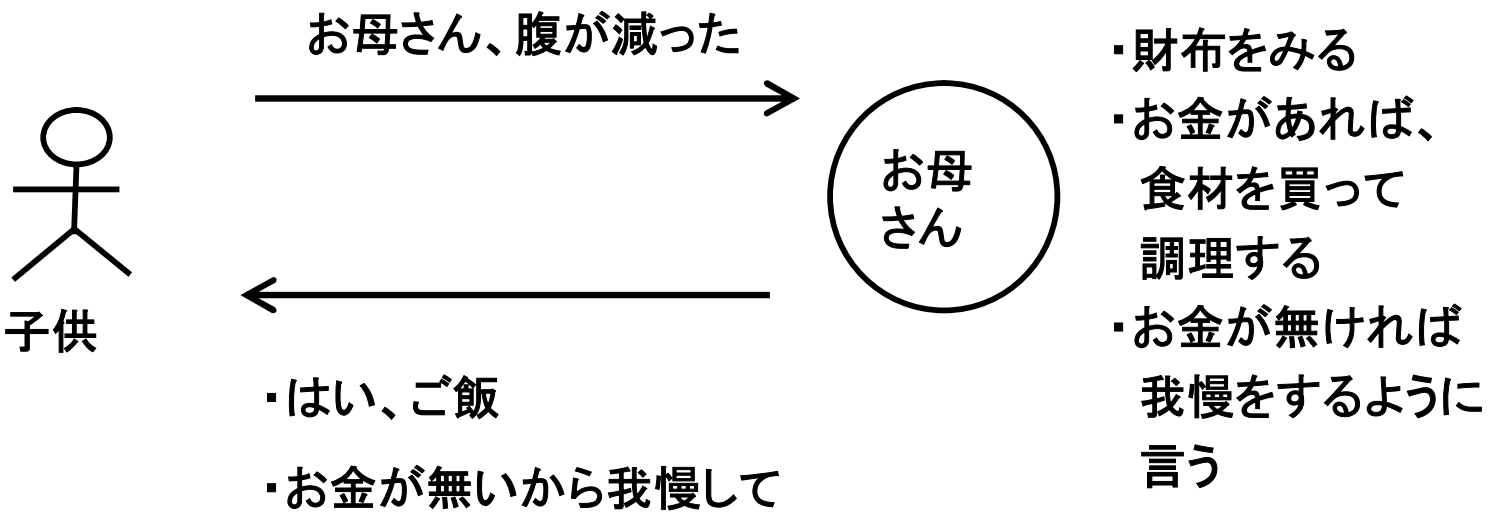
- 現実の世界は、いろいろなモノが役割を分担しながら機能を果たしている。
- 自分でできることは、自分でやり、自分の範囲外の仕事は人に任せて、結果だけをもらう

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

②オブジェクト指向の考え方

- 現実のモノを中心とした考え方
- 人間の発想、活動に近い考え方で
  システムのモデリングを行うもの
- 目的志向の即物的な考え方



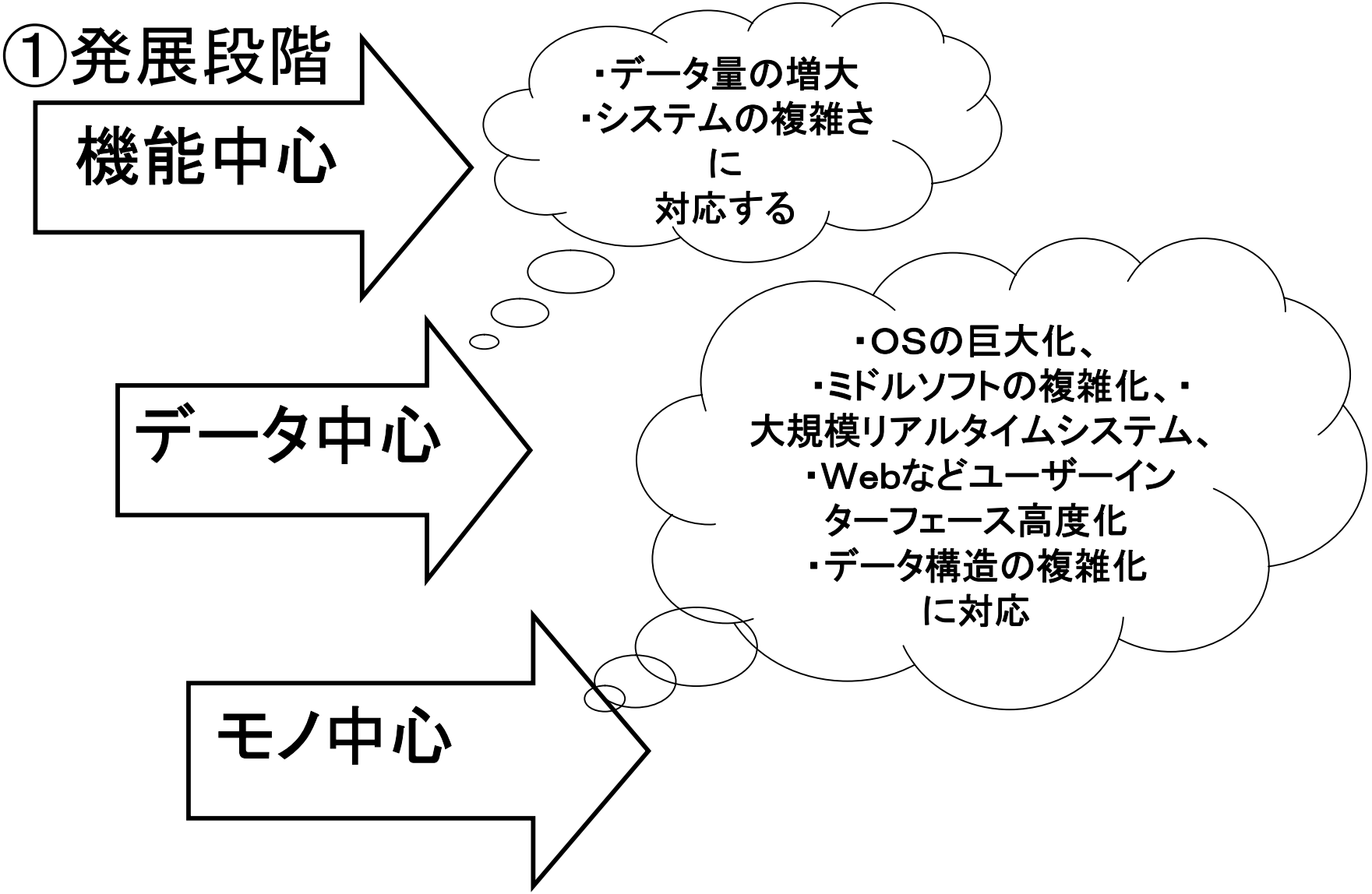
１．２　オブジェクト指向のねらい

「システム開発を現実の世界に近づけることにより、
システム開発のシンプル化、効率化を実現する」

・現実に近い形でモデルを作り、ソフトウェアの
構造の複雑さを軽減する。
・上流工程から下流工程まで、成果物に一貫性
を持たせる。
・工業製品と同様に、部品を組合わせて、
開発することで、効率化を進める。
・ソフト部品化が進み、部品再利用が進む。
・保守の局所化ができ、保守が容易になる。

１．３　オブジェクト指向成立の経緯

①発展段階



②経緯

・ｵﾌﾞｼﾞｪｸﾄ指向言語の開発が発端

－1968、ﾉﾙｳｪｰ・DohlI氏のSIMULA67(ｸﾗｽ概念)

－1970、Zerox・ﾊﾟﾛｱﾙﾄ研・Key氏のSmalltalk

（人にやさしい、子供でも理解・操作できる）

・ﾌﾟﾛｸﾞﾗﾑ作成効率化のために、上流の技法を開拓

－Rational　Software社・Booch氏の設計手法

（論理ﾓﾃﾞﾙ・動的ﾓﾃﾞﾙ・物理ﾓﾃﾞﾙ）

－GE・Rumbaugh氏の分析手法

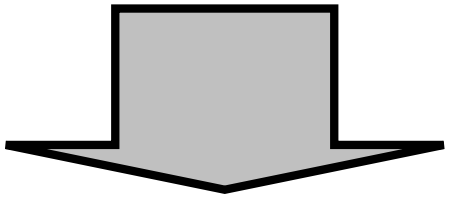
（ｵﾌﾞｼﾞｪｸﾄﾓﾃﾞﾙ、動的ﾓﾃﾞﾙ、機能ﾓﾃﾞﾙ、

ﾕｰｽｹｰｽ）

－Shlaer&Mellorの大規模リアルタイムシステム技法
（情報モデル・状態モデル・プロセスモデル）

－Cord&Yourdonの分析・設計技法
（多重レイヤモデル、多重コンポーネントモデル)

－Jacobsonのビジョン、ユースケース



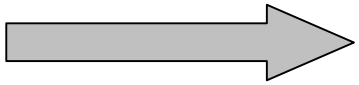
－ Booch、Rumbaugh、Jacobsonによる
モデリング統一化への取り組み
・統一分析・設計の方法論---まとまらず
・1997、統一モデリング言語に限定して統一化

③ 統一モデリング言語（UML）

- Unified Modeling Language
- ｵﾌﾞｼﾞｪｸﾄ指向のﾓﾃﾞﾘﾝｸﾞをグラフィカルに記述する表記方法である。
- 従来の適用範囲
  - ー機器・装置の制御系ｼｽﾃﾑ
  - ーﾏｲｺﾝなどの埋め込みｼｽﾃﾑ
  - ー大規模リアルタイムシステム
  - ー基本ソフト（OS、およびNW、GUI、DBMSなどのミドルウェア）
  - ー分散、Webｼｽﾃﾑなどの適用業務
- 最近の動向
  - ーシステム設計方式の統一に適用され始めた。

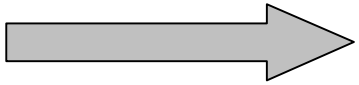
④過去の手法の継承（例示）

- データモデリング
  - ーデータ正規化
  - ーERモデリング
- リアルタイム手法
  - ーイベントドリブン
  - ー状態遷移
- ﾌﾟﾛｸﾞﾗﾑ分割手法
  - ーモジュール強度
  - ーモジュール結合度



・オブジェクト
クラスの
概念に取り入れ

・メッセージ、状態
遷移の考え方に
取り入れ



・クラスの大きさ、
クラス間の境界の
決定に取り入れ

⑤プログラミング言語の系譜

1960	ALGOL	数値計算用の言語
1967	Simula67	クラス概念
1972	Smalltalk	初めてのｵﾌﾞｼﾞｪｸﾄ指向
1985	C＋＋	ｵﾌﾞｼﾞｪｸﾄ指向、Cの改良版
1995	Java	ｵﾌﾞｼﾞｪｸﾄ指向
2000	C#	ｵﾌﾞｼﾞｪｸﾄ指向、. NET対応

１．４　適用が遅れた原因

①ユーザー企業の既存ソフト資産の存在
　　－既存のソフト資産を最大限使いたい。

②適用範囲が限られていた
　　－機器制御、ＯＳ、ミドルソフト中心

③教育の不足
　　－人によるレベル差が非常に大きい

④開発プロセスが未確立
　　－追加型（ｲﾝｸﾘﾒﾝﾀﾙ）、ｳｫｰﾀﾌｫｰﾙ型
　　－ｺﾝﾎﾟｰﾈﾝﾄとアプリの平行開発
　　－モデルの表記方法が先行してきた
　　－目的に応じたクラス定義が難解

⑤ コトバの混乱

・新しいコトバ、概念が多い
・改革と改善が混在

- クラス
- インスタンス
- オブジェクト
- スーパークラス
- サブクラス
- パッケージ

- アクター
- ユースケース
- コラボレーション
- 状態遷移
- 相互作用
- アクティビティ
- コンポーネント
- 配置

- メッセージ
- イベント
- メッセージパッシング

- アトリビュート
- 属性
- データ
- 情報
- 操作
- メソッド
- 振る舞い
- 手続き
- 処理
- サービス

- 抽象化
- ポリモフィズム
- 多態性
- イントロスペクション
- 差分プログラミング

- 情報隠蔽
- インヘリタンス
- 階層化
- 関連
- 継承、汎化／特化
- 集約
- 所有
- 多重度
- オーバーロード
- is a、 has a

- アーキテクチャー
- フレームワーク
- クラスライブラリー
- ステレオタイプ
- メソッド
- デザインパターン
- メタパターン
- コンポーネントウェア

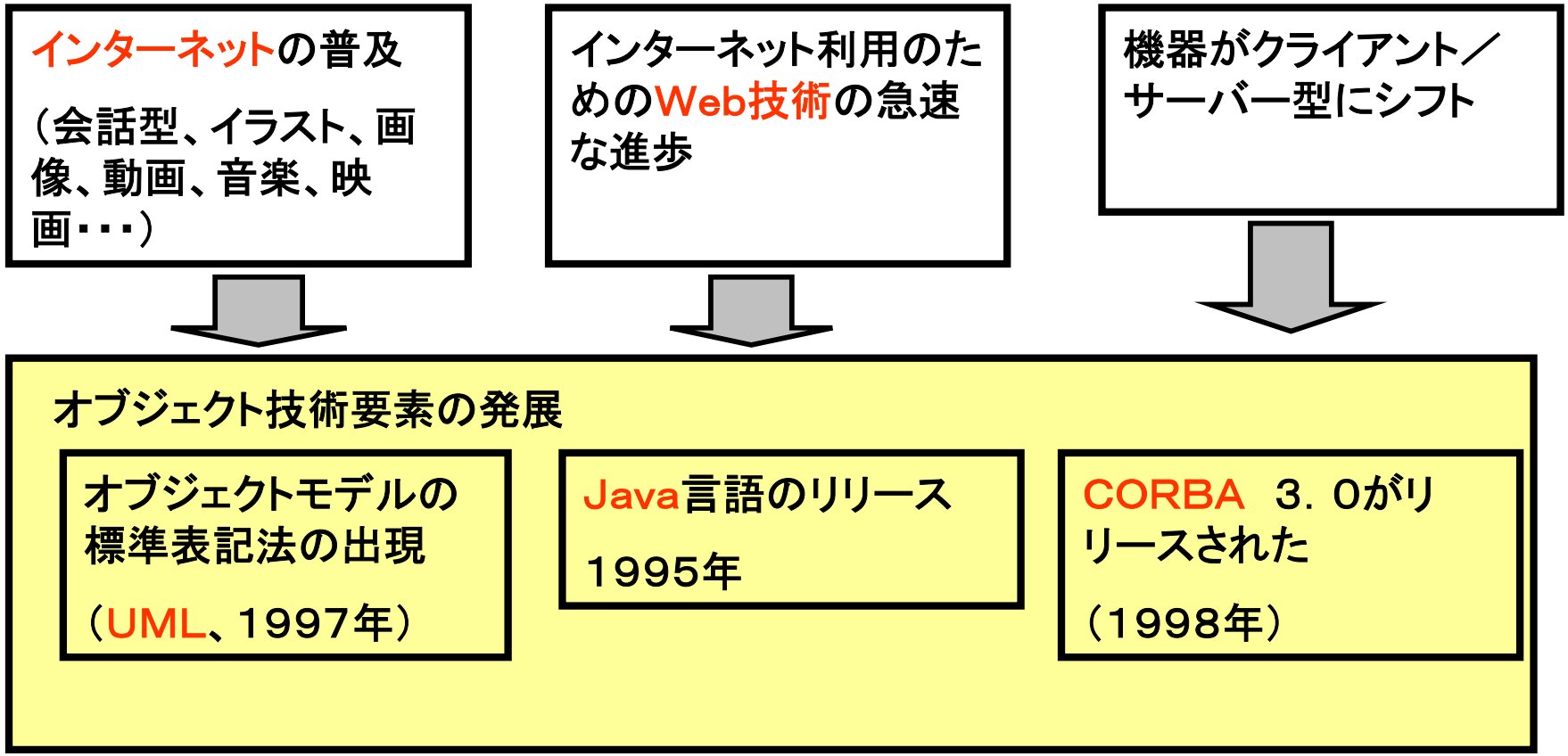
- インクリメンタル開発
- 追加型開発
- スパイラル型開発
- OOA、OOD、OOP
- 静的、動的、振る舞い、データモデル

１．５　ビジネスモデルへの適用が進む理由

・インターネットの普及により、Webアプリケーションが増大した

・オブジェクト指向の言語（Java）が普及してきた

・拡大するC／S型の分散処理領域にも、オブジェクト指向の標準化が進んだ



２．オブジェクト指向の基礎概念

２．１　オブジェクトの基本概念

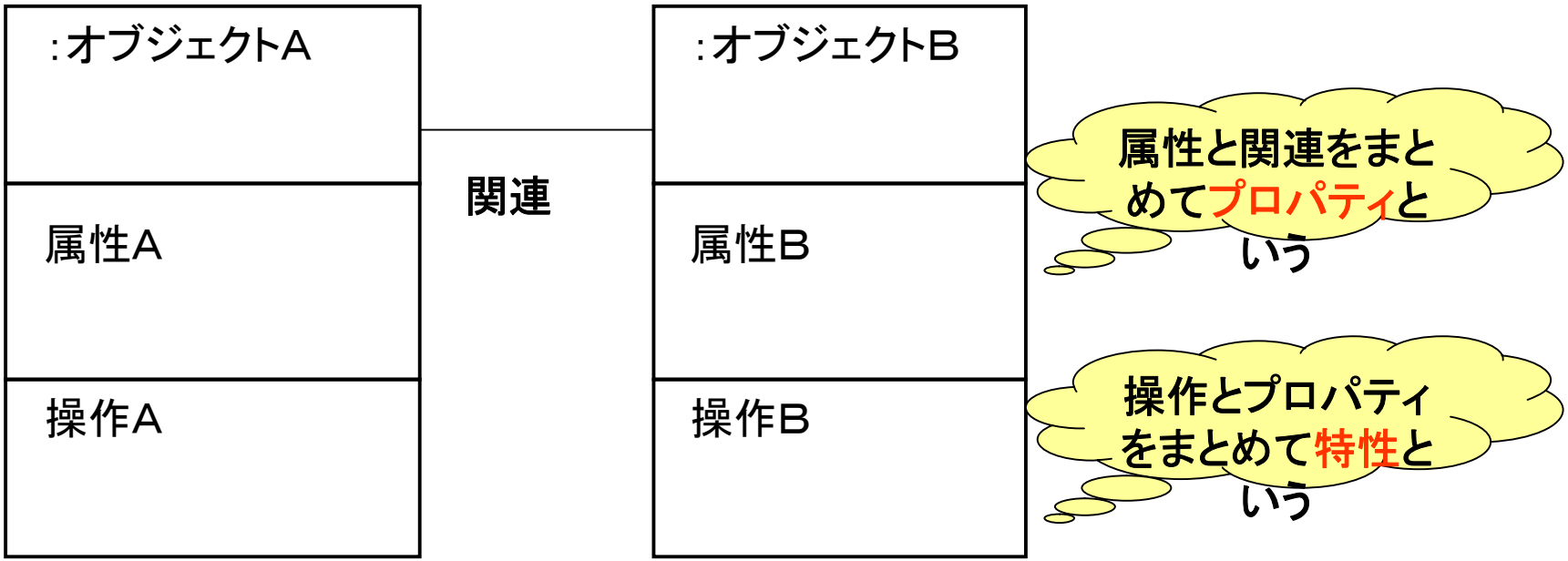
①オブジェクト、クラス、インスタンス

・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

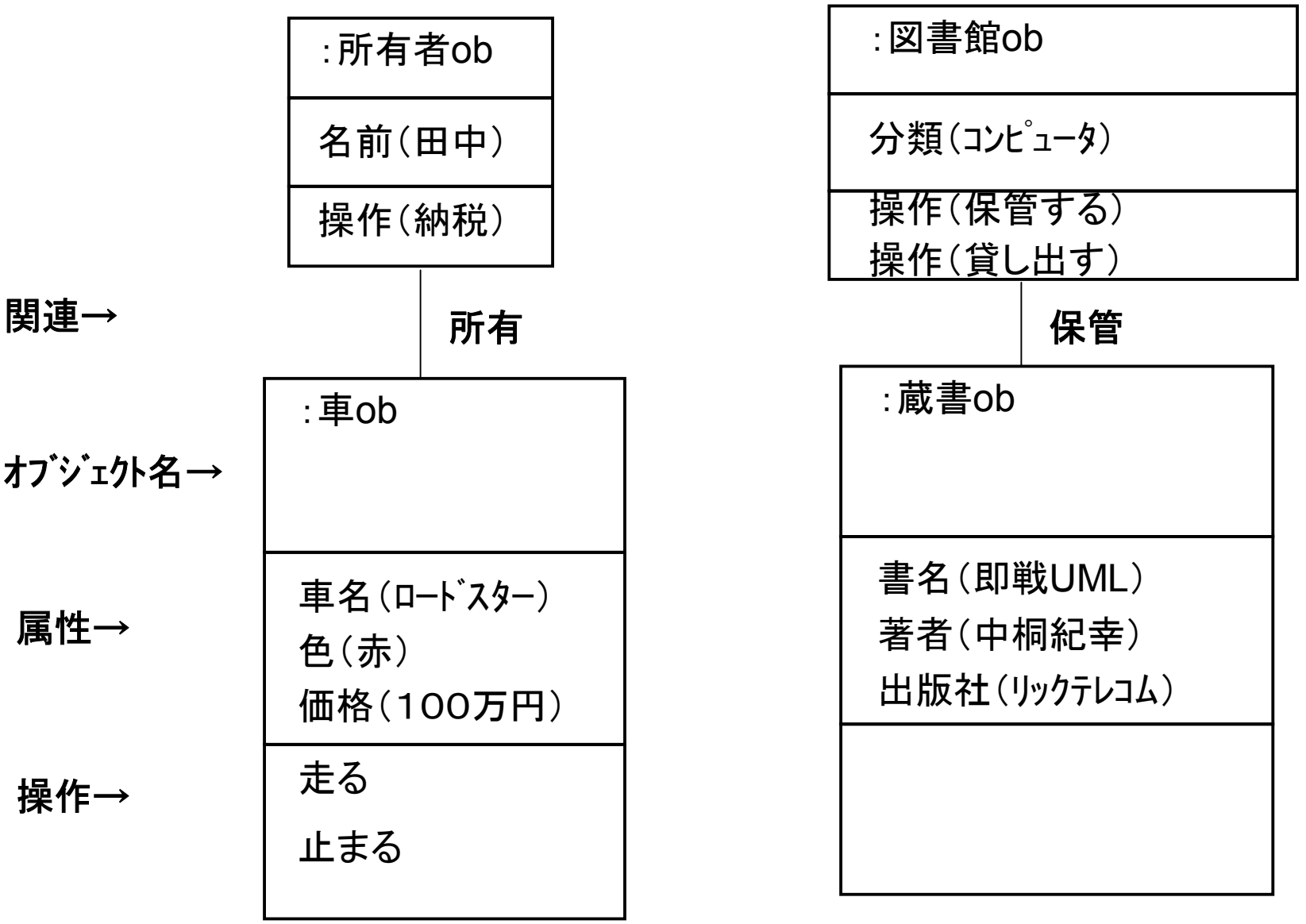
－人　　：社員、顧客、株主、学生、先生・・・
－組織　：自社組織、特約店・・・
－場所　：本店、支店、販売地区、工場、倉庫、
　　店舗、配送拠点・・・
－物　　：商品、製品、部品、材料、設備、
　　機器、金、伝票、帳簿・・・
－事象　：受注、発注、入庫、入金、売掛・・・
－概念　：業務、スキル、目標、法規・・・

・オブジェクトの特徴

－固有の姿、形、性質などの属性を持つ
－固有の振る舞い（＝操作）を持つ
－他のオブジェクトとの間に固有の関係を持つ（＝関連）



（例示）


:所有者ob
名前（田中）
操作（納税）
関連→
所有
:車ob
オブジェクト名→
属性→
車名（ロードスター）
色（赤）
価格（100万円）
操作→
走る
止まる
:図書館ob
分類（コンピュータ）
操作（保管する）
操作（貸し出す）
保管
:蔵書ob
書名（即戦UML）
著者（中桐紀幸）
出版社（リックテレコム）

- 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。

・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
　　ーメッセージドリブン

} カプセル化

・部品化、再利用化が容易になる。

２．２　クラス間の関係に関する基本概念

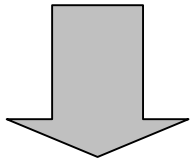
①　継承、汎化、特化

- 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）

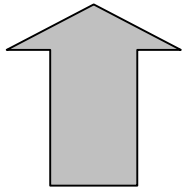
・このとき、下位クラスは、上位クラスの特性を引継ぐことができる。
（継承（インヘリタンス）という）
（is－a関係、a－kind－of関係ともいう）

・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。

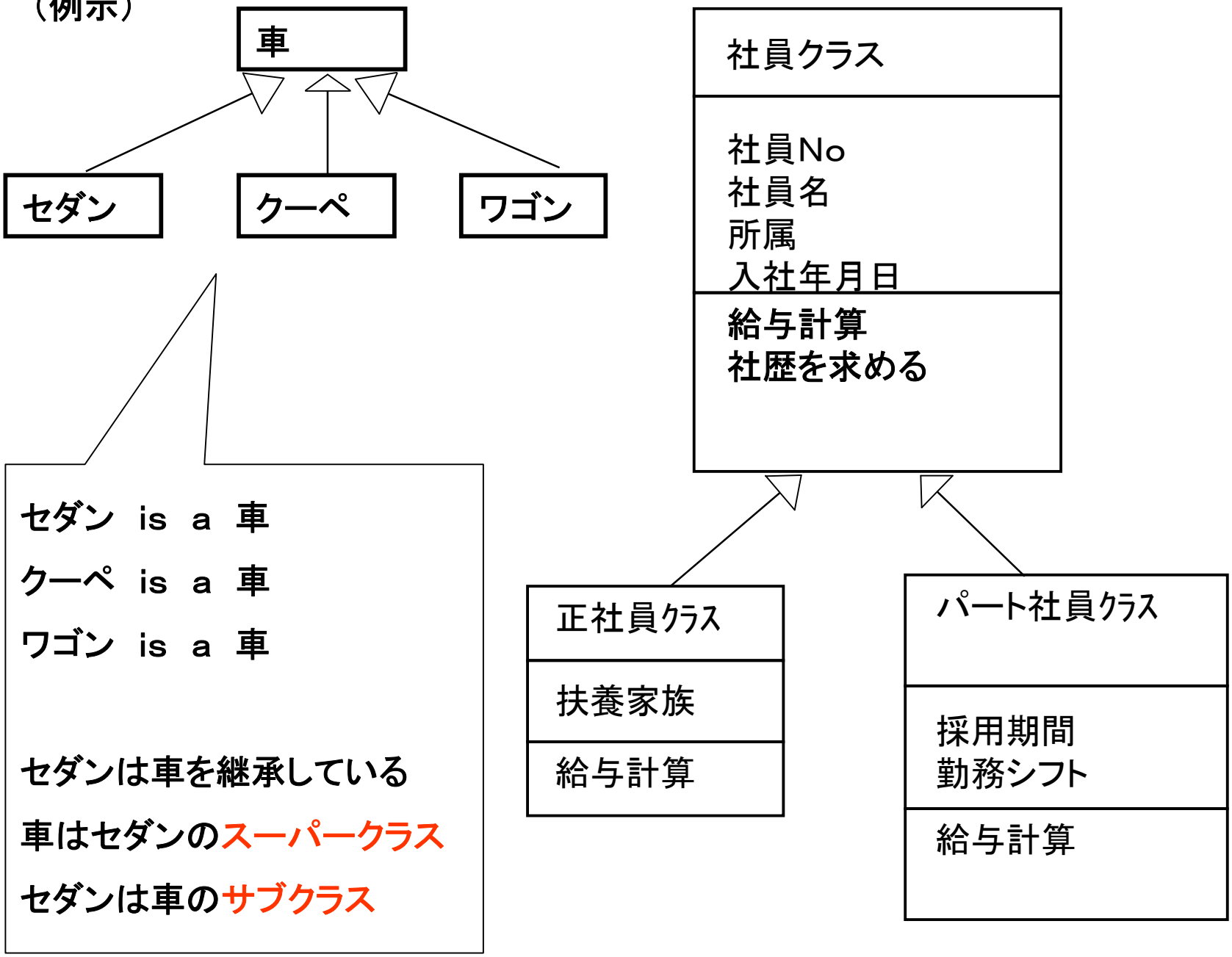


・汎化－－定義済みのクラスの共通的な特性に着目し、共通部分を取り出し、上位クラスとして導き出すこと。



・変更に対して柔軟、重複部分の開発を削減可能。

（例示）


車
セダン
クーペ
ワゴン
セダン is a 車
クーペ is a 車
ワゴン is a 車
セダンは車を継承している
車はセダンのスーパークラス
セダンは車のサブクラス
社員クラス
社員No
社員名
所属
入社年月日
給与計算
社歴を求める
正社員ｸﾗｽ
扶養家族
給与計算
パート社員ｸﾗｽ
採用期間
勤務シフト
給与計算

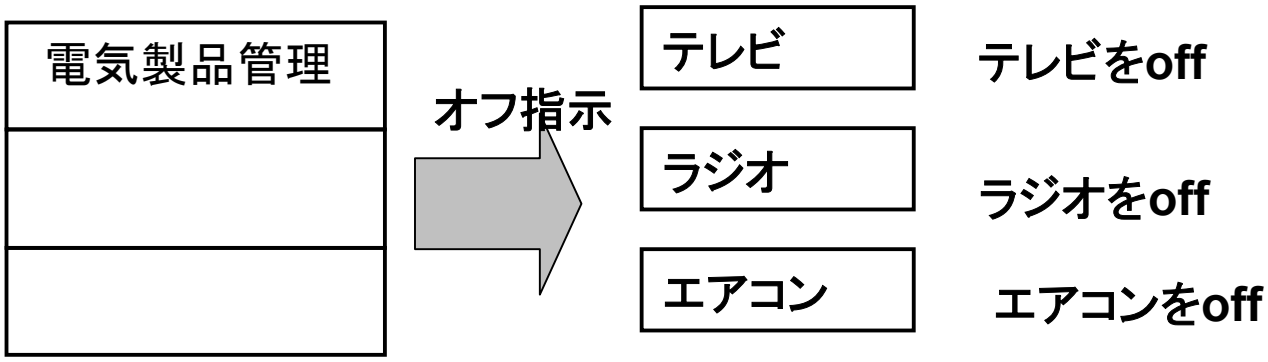
②多態性

- 継承によって、下位クラスは、上位クラスからの同じメッセージに対して、それぞれ下位クラスで定義した、異なる操作を実行できる。
- 多様性、多相性とも言う。

  ﾎﾟﾘﾓｱﾌｨｽﾞﾑ（poly多様な、morphism形態を持つ状態）

・ソフトウェアの再利用性が高まる。

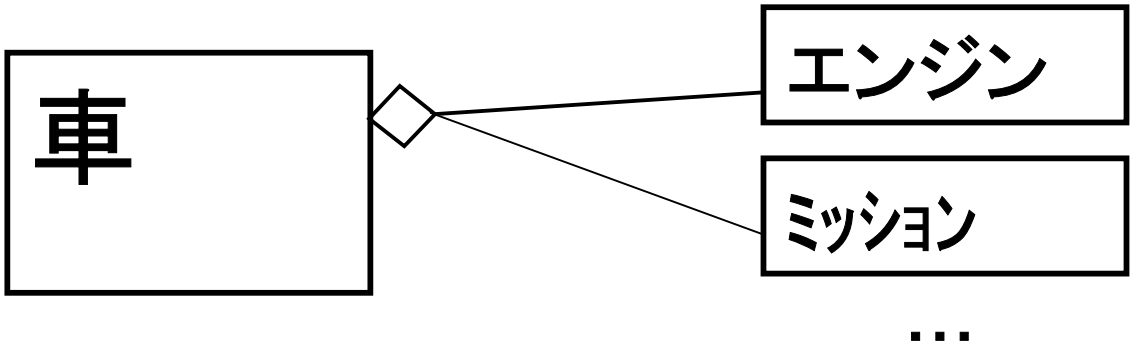
（例示）



③ 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）



３．オブジェクト指向の手法

３．１　有効なオブジェクト技術

①ステレオタイプ

－URLに定義されているモデリング要素の
　意味を拡張する手法。
－これによりULMの利用者が、独自に
　意味を与えることができる。
－《　　》で囲むのが表記法。

②フレームワーク

－関連性の強いクラスの集合であり、予めテンプ
　レートとして用意する。
－追加が必要なｸﾗｽを部品として組込むことが出来る。
－GUIのフレームワークが良く作成されている。

③ デザイン・パターン

- フレームワークを効果的に構築する手法である。
- クラスの大きさをガイドしてくれる
- Gamma氏含めて４名（Gang of Four）が、さまざまなシステムを調査して、GUIのパターンを発見。（生成、構造、振舞いに3分類）
- Facade（ファサード、玄関）は、構造に関するパターンであり、サブシステムの入口にクラスを設けるやりかた。これによりサブシステムを構成するクラスは、外部からは隠蔽できる。

④MVC（モデル／ビュー／コントロール）

－変更に強い柔軟なクラス設計の技法

－１つのクラスを、モデル・ビュー・コントローラーに分割する。

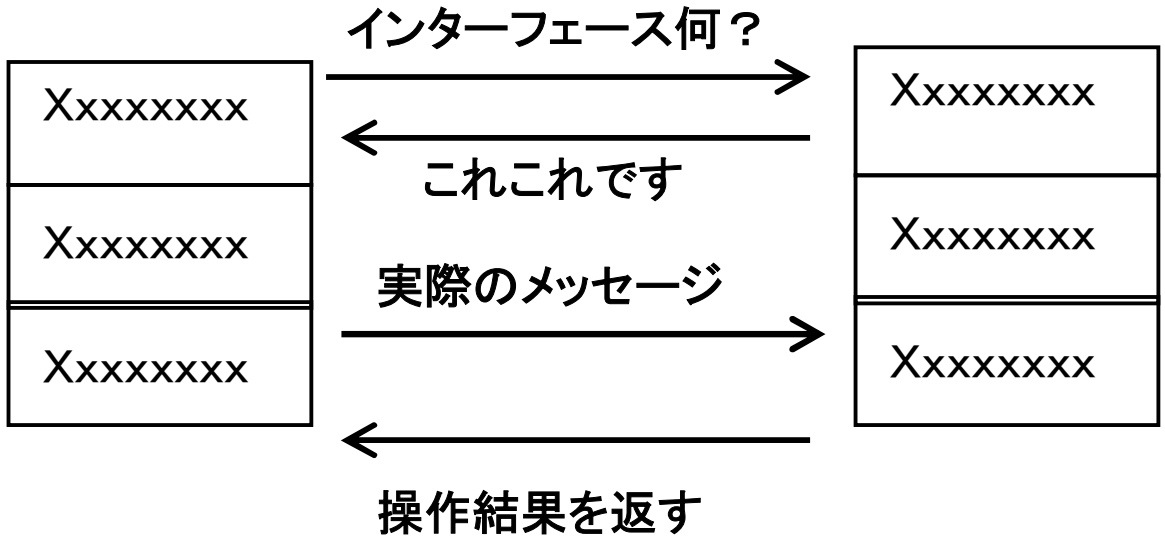
－対話型アプリケーションでは、
モデル（本質的なデータを管理するクラス）
ビュー（表示装置とのインターフェースに責任を持つ）
コントローラー（I／O装置からのイベントの受付）

－３階層構造のアプリでは、DB管理、ビジネスロジック、ユーザービューに分割する。

⑤ｲﾝﾄﾛｽﾍﾟｸｼｮﾝ

－あるクラスが、自分自身のことを把握できる
　機能である。

－柔軟なシステム化が可能となる



⑥分散オブジェクト技術

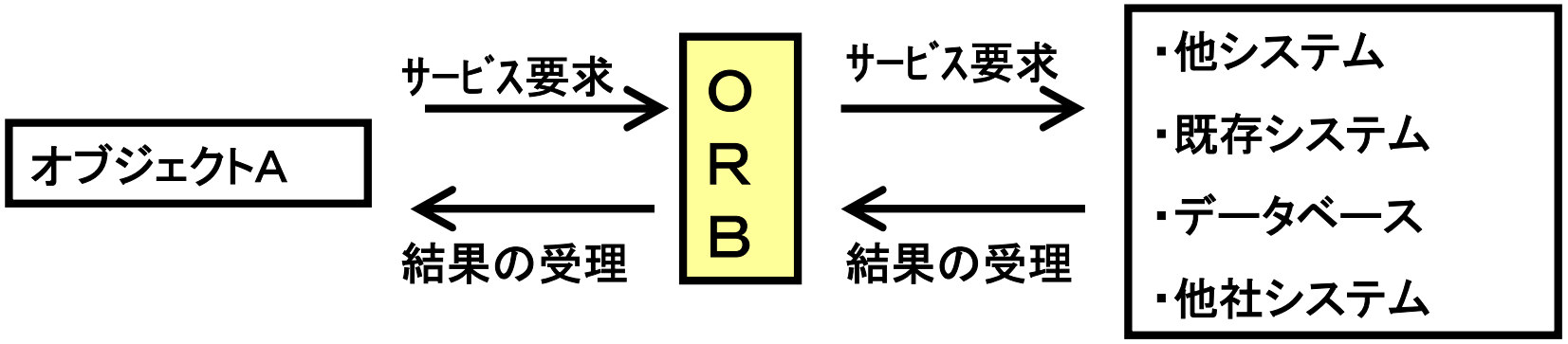
－異機種が混在するネットワーク環境下で、
分散・遠隔配置されたオブジェクト（クラス）を
相互に連結して通信・使用できるように
するためのシステム統合の標準化技術である。

－今後のｺﾝﾋﾟｭｰﾀﾃｸﾉﾛｼﾞｰのキー技術

－OMGで標準規格としてCORBAを作成。

（Common Object Request Broker Architecture）
（IIOP＝Internet Inter ORB Protocol）

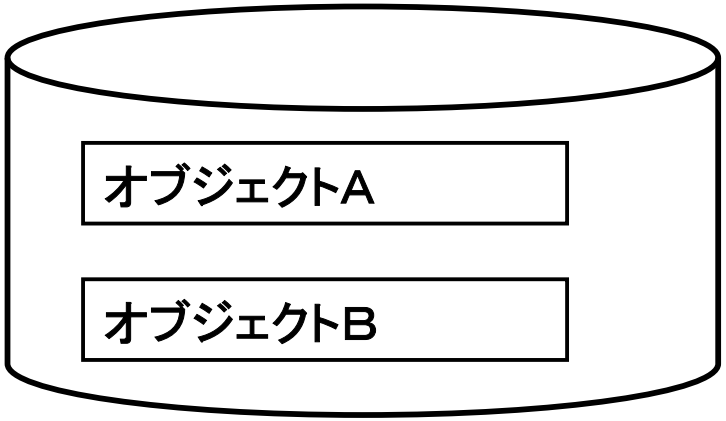
－ﾏｲｸﾛｿﾌﾄは、独自規格を用意（COM/DCOM）



⑦オブジェクト指向データベース

・２種類のＤＢＭＳがある

－純粋な形のＤＢＭＳ（ＯＯＤＢ）



オブジェクトがそのまま入る

－既存のリレーショナルデータベースの機能追加により、オブジェクトが格納出来る（ＯＲＤＢ）

・普及は、いまから。

３．２　開発のプロセスに於ける手法

①アプリケーション検討・開発の対象

- オブジェクト指向のシステム分析／設計
- 〃　　プログラミング
- 〃　　データベース
- 〃　　ユーザーインターフェース

②モデル定義の手法

- システムの外部的な視点
  ー要求を分析し、ユーザー要求を満足する
  ーユースケース（機能モデル）を分析する
- システムの内部的な視点
  ー静的な構造を明らかにする（＝ｵﾌﾞｼﾞｪｸﾄﾓﾃﾞﾙ）
  （クラスの抽出、クラス間の関係を抽出）
  ー動的な構造を明らかにする（＝相互作用ﾓﾃﾞﾙ）
  （クラスの使用シーケンス定義、状態遷移定義）
- システムの物理的な視点
  ーソフトウェア、ハードウェア、NWの配置などの
  物理的な構成を明らかにする。

③　DOAなど構造化技法との工程の差異

・要件定義の工程は、DOAと同じである。

・外部設計以降は、オブジェクトの設計（つまりその抽象概念のクラス設計）が中心

工程	OO（ｵﾌﾞｼﾞｪｸﾄ中心）	DOA（ﾃﾞｰﾀ中心）
現状調査・要求検討	要求分析 （一部OOA手法）	要件定義
外部設計	OO分析（OOA）	外部設計
内部設計	OO設計（OOD）	内部設計
製造・テスト	製造・テスト	製造・テスト
統合テスト、移行	統合テスト、移行	統合テスト、移行
運用・保守	運用・保守	運用・保守

4. オブジェクト指向の開発工程

①要件定義フェーズ

・要件定義のためのOO専用の手法は少ない。
　－せいぜいユースケース分析がある程度。
　－現状調査、要求分析はDOAと同様に
　　やる必要がある。
　－インクリメンタル開発（段階的）が可能なのは
　　小規模で自明の内容を多く含んでいる場合。
・対象の現状を、オブジェクト指向でモデリングして
　おくのは、改善型のシステム開発の場合は有効。
・含めるべき検討項目は以下の通り

－システム化の背景、目的、目標レベル
－システム機能（ユースケース図／ユースケース記述）
－求められる品質レベル（応答時間・・・）
－アーキテクチャー（H／W階層、OS、ﾐﾄﾞﾙｿﾌﾄ、GUI、
　　DBMS、通信形態・・・）
－予算、開発期間、効果、実現機能の優先順位など

②分析フェーズ（外部設計）

- 要件定義で作成した機能定義（ユースケース図、ユースケース記述）をもとに、オブジェクトモデル（静的構造）を作成する。
  - ―クラス取出し、属性定義、操作定義、クラス間の関係定義、クラス図作成
  - ―サブシステムに分割（パッケージ図）
- 次いで動的モデルを作成する。
  - ―ユーザーとクラスとの関係を分析し、シーケンス図を作成。
- まとめとして、属性、操作の用語集を作成する。

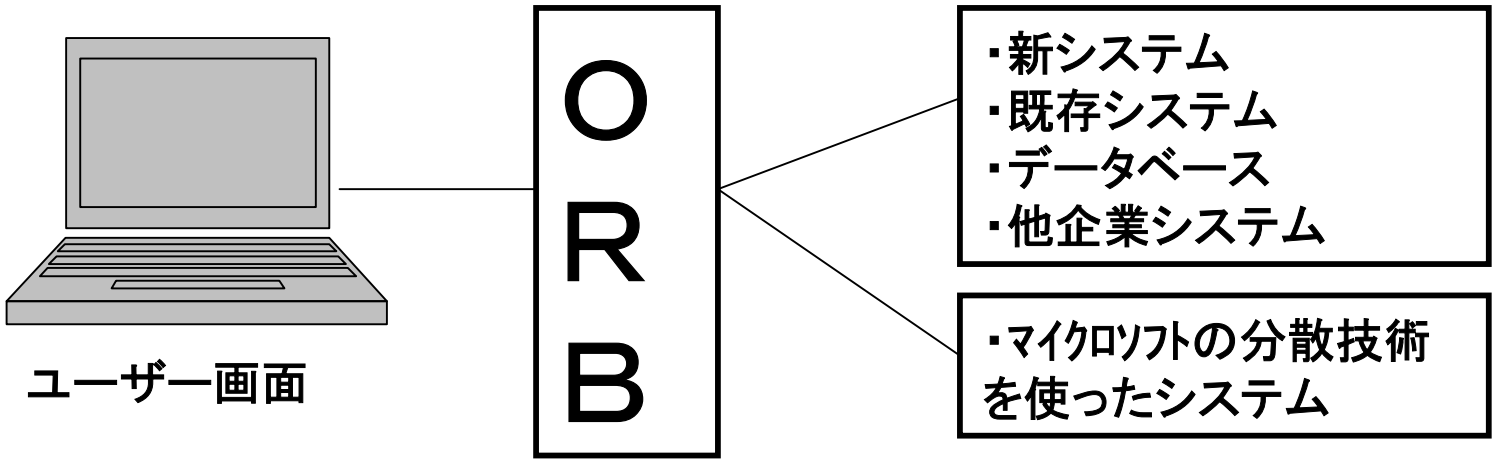
③設計フェーズ（内部設計）

- システムの実装環境での、システム構造を決定する。
- 分析フェーズで得られたモデルを詳細化する。
- 設計書の作成
  - ー実装環境での、ｸﾗｽ配置、ﾃﾞｰﾀｽﾄｱ配置を決定。
    (配置ﾋﾞｭｰ)
  - ー画面、帳票の設計を最終決定。
- オブジェクトモデルの完成（ クラス特性の完成 ）
  - ーｲﾝﾀｰﾌｪｰｽｸﾗｽ、ﾃﾞｰﾀｽﾄｱｸﾗｽの追加
  - ーｸﾗｽ間の関連の見直し追加（継承、集約）
  - ークラス図、クラス仕様書の作成
- 動的モデルの見直し
  - ー追加したクラスをシーケンス図に反映
  - ーオブジェクトの生成～消滅を明確化
- 再利用の検討
  - ー再利用のための設計見直し
    （フレームワーク／MVCの検討／部品の定義）
  - ーコンポーネントの適用

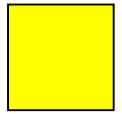
5.企業におけるオブジェクト指向

- 顧客とのコンタクト強化
  - ーWebｼｽﾃﾑ、顧客情報、ｺｰﾙｾﾝﾀｰ、
  - ーDWH、データマイニング
- 商品開発の期間短縮、品質向上
  - ーデジタルエンジニアリング、
  - ーバーチャル試作、生産準備
  - ーナレッジベース
- 生産、ロジスティックスの最適ビジネスモデル
  - ーBTO、ﾘｱﾙﾀｲﾑMRP、ﾜｰﾙﾄﾞﾜｲﾄﾞ最適
- 企業内情報システムの統合
  - ーｲﾝﾄﾗﾈｯﾄ、ｺﾗﾎﾞﾚｰｼｮﾝ、ﾜｰｸﾌﾛｰ
  - ーﾕﾋﾞｷﾀｽ、ﾓﾊﾞｲﾙ、Web会議

- 分散オブジェクト技術により、新システムだけでなく、既存システムでも、オブジェクト指向を生かして使える時代になった。
  - ー利用者が同じブラウザー操作により、企業内システムを使いたい。
  - ー既存システムはそのままで、ORBを中継すれば可能である。



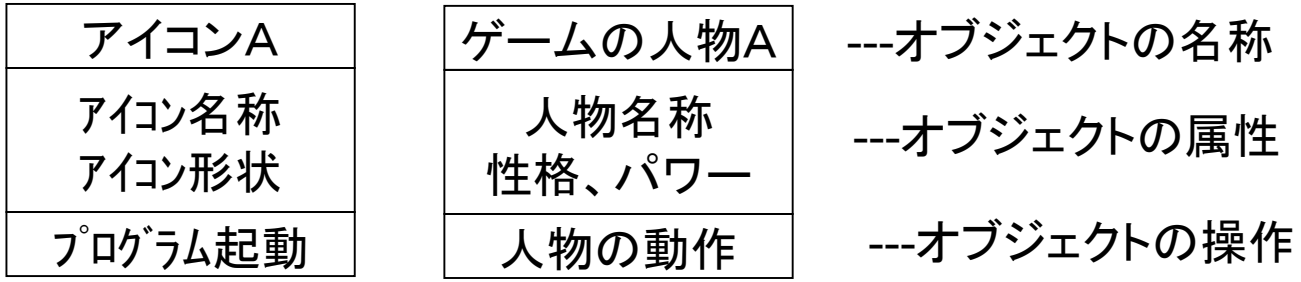
補足：システム化の工程と技法



- 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））

補足: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



- 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・・・）

６．まとめとレポート課題

- 重要項目

ーｵﾌﾞｼﾞｪｸﾄ指向の基本概念（ｵﾌﾞｼﾞｪｸﾄ、関係）
ーオブジェクト指向の手法と開発プロセス

- レポート課題（A4x1、２枚）

①オブジェクト指向の基本概念を説明しなさい。
②自分がやってみたい対象システムの
　クラスを書き出してください。

　期限　　次回の授業開始時点
　提出　　レポート用紙またはメール

７．参考書、参照URL

- 情報処理学会、「情報処理１９９４年、５月号」
  ―特集「オブジェクト指向分析・設計」
- 中桐紀幸、「即戦UMLモデリング」(ﾘｯｸﾃﾚｺﾑ)
- 布川薫ほか「ｱﾌﾟﾘｹｰｼｮﾝ開発技術」(ﾘｯｸﾃﾚｺﾑ)
- 別所義夫「これで使えるｵﾌﾞｼﾞｪｸﾄ指向」（電気通信協会）
- 磯田定宏「ｵﾌﾞｼﾞｪｸﾄ指向ﾓﾃﾞﾘﾝｸﾞ」（コロナ社）
- 落水浩一郎「ｵﾌﾞｼﾞｪｸﾄﾓﾃﾞﾘﾝｸﾞ」（星雲社）
- 酒井博敬「オブジェクト指向入門」（オーム社）
- (株)永和「ｵﾌﾞｼﾞｪｸﾄﾊﾝﾄﾞﾌﾞｯｸ」（ﾋﾟｱｿﾝｴﾃﾞｭｹｰｼｮﾝ）
- 河合昭男「図解、ｵﾌﾞｼﾞｪｸﾄ指向がわかる」（技術評論社）
- 情報処理学会「情報処理ﾊﾝﾄﾞﾌﾞｯｸ」（オーム社）

参考になるURL

- http://www.ogis-ri.co.jp/otc/hiroba/technical/oodnote/serial1/oodnote1.html オブジェクト指向の設計、Javaソースコード付き
- http://www.ogis-ri.co.jp/otc/hiroba/index.html オブジェクト広場
- http://ObjectClub.esm.co.jp/ UML、Java
- http://www.rational.co.jp/uml/ UML
- http://www.zzz.or.jp/~iizuka/EOO/index.html
  OOのリンク集
- http://www1.u-netsurf.ne.jp/~kitada/3H/index.htm
  OOの概念説明
- http://www.njk.co.jp/otg/ OO全般