オブジェクト指向
システム分析／設計
入門：実践的アドバイス

- 佐原伸
  - E-Mail
    - sahara@sra.co.jp
  - URL
    - http://www.sra.co.jp/people/sahara
- ＳＲＡオブジェクト指向グループ
  - URL
    - http://www.sra.co.jp/smalltalk

目次

オブジェクト指向の考え方
オブジェクト指向方法論
オブジェクト指向分析
オブジェクト指向設計
オブジェクト指向プログラミング
オブジェクト指向プロジェクトの管理
オブジェクト指向の教育

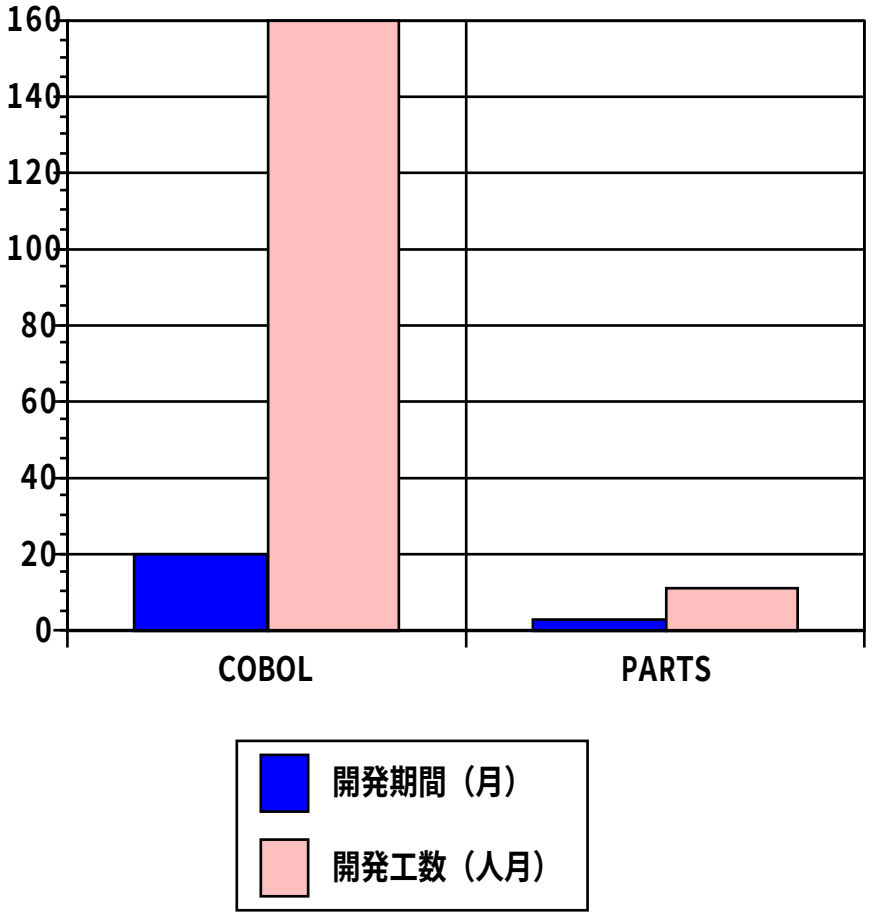
オブジェクト指向の考え方

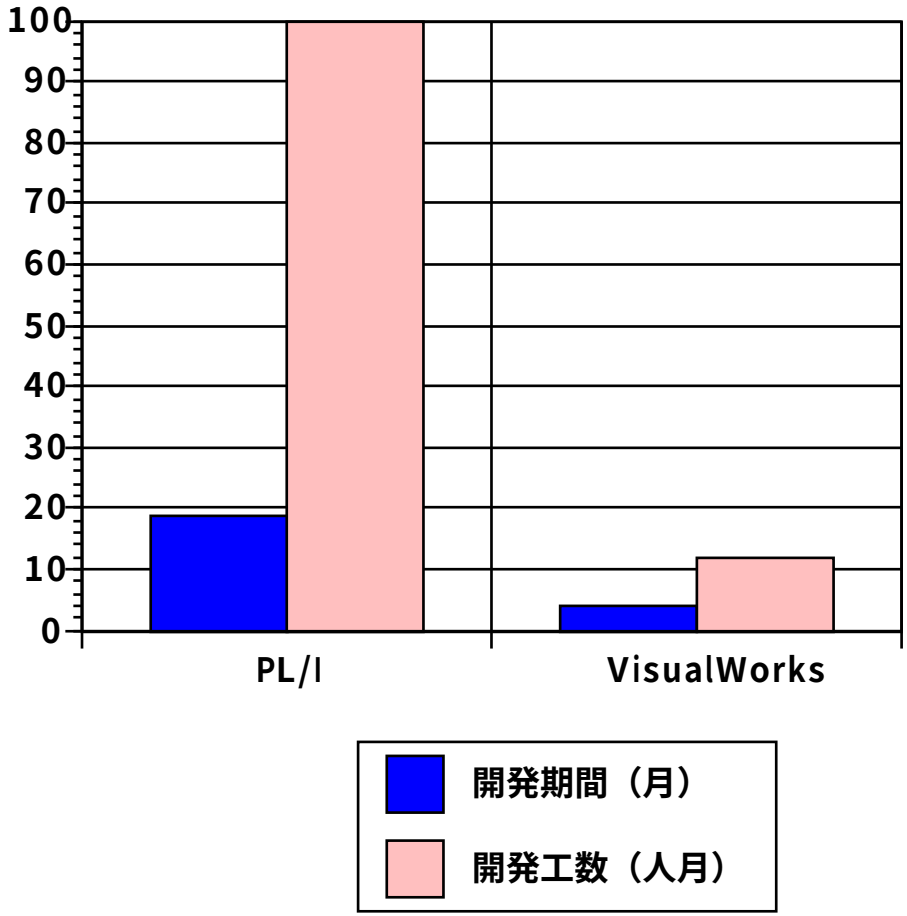
- なぜオブジェクト指向なのですか？
- オブジェクト指向は、実際に使われているのですか？
- オブジェクト指向の基本概念
- オブジェクト指向の位置付け

なぜオブジェクト指向なのですか？

- 経済的理由
- ソフトウェア工学的理由

経済的理由


160
140
120
100
80
60
40
20
0
COBOL
PARTS
開発期間（月）
開発工数（人月）


100
90
80
70
60
50
40
30
20
10
0
PL/I
VisualWorks
開発期間（月）
開発工数（人月）

ソフトウェア工学的理由

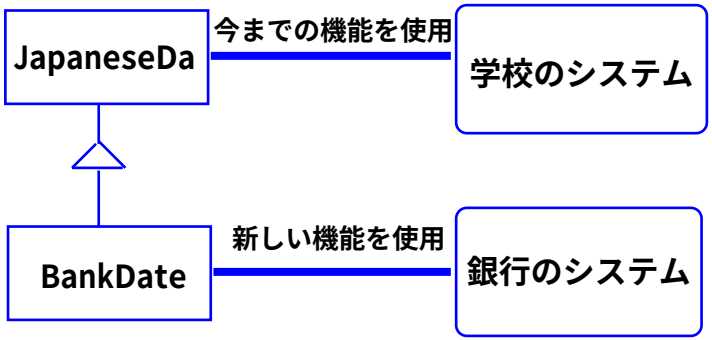
- ソフトウェア工学からの要求
  - 外的品質要因
  - モジュール性の原則
  - 再利用可能なモジュール構造の要件

外的品質要因

- 正確さ
  - 要求された通りに仕事を行う能力
- 頑丈さ
  - 異常な状態においても機能する能力
- 拡張性
  - 仕様の変更に容易に適応できる能力
- 再利用性
  - 新しい応用にどの程度再利用できるかを示すもの
- 互換性
  - ソフトウェア相互の組み合わせやすさ

モジュール性の原則

- 言語としてのモジュール単位
- 少ないインタフェース
- 小さいインタフェース
- 明示的なインタフェース
- 情報隠蔽
- 開放／閉鎖の両立
  - 以下を同時に満たさなければならない
    - モジュールが拡張可能である（開放）
    - モジュールが他のモジュールから使用できる（閉鎖）



再利用可能なモジュール構造の要件

- 型の変化に対応できる
- データ構造とアルゴリズムの変化に対応できる
- 関連した操作がまとまって定義されている
- 顧客モジュールが実現方法を知ることなく操作を要求することができる
  - 今までの技術ではうまく解決できない
  - 後述する動的束縛で解決できる
- 実現方法の間の共通部分がうまくまとめられている
  - サブグループ間の共通性をうまく記述できるか
    - 表検索の場合では、順次形式の表がひとつのサブグループになる
      - 順次配列、連結リスト、順次ファイルの共通点をうまくまとめられるか？

オブジェクト指向は、実際に使われているのですか？

- マルチメディア
  - アトランタ・オリンピック
  - DODの空母搭載基地情報システム
  - 地図情報システム
- ビジネス
  - 銀行
  - 在庫管理
- リアルタイム制御システム
  - 航空管制システム、衛星制御システム、中国の鉄道システム（マカオ国連大学）

オブジェクト指向の基本概念

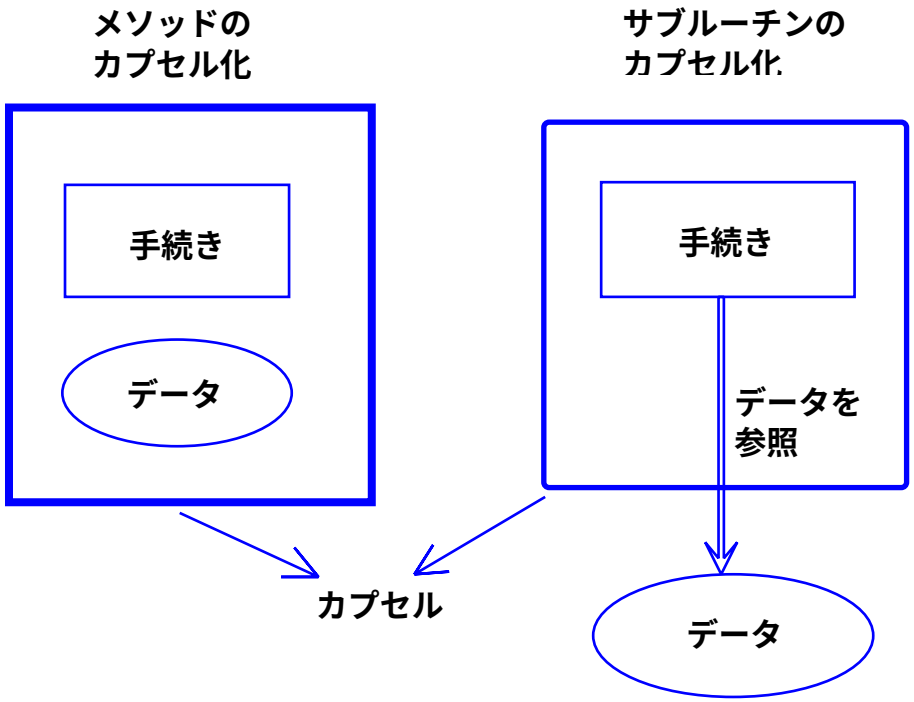
- クラスとインスタンス
- カプセル化
- メッセージ
- 多相と動的束縛
- 継承
- 自動メモリ管理

クラスとインスタンス

- なぜオブジェクトをクラスとインスタンスに分けるのですか？
  - 単なる都合
    - クラスにインスタンス共通の性質を記述するため
    - インスタンスベースのオブジェクト指向言語もある
- メタクラスとは何ですか？
  - クラスの性質を記述するためのクラス
    - メタクラスから見ればクラスはインスタンス

カプセル化

- カプセル化しない方が効率がよいのではないですか？
  - システム全体の効率に関係する操作は全体の２～３％
    - 分析／設計段階からミクロの効率化を考えると、マクロ的に見れば非効率なシステムになる
- カプセル化はサブルーチンで十分ではありませんか？
  - サブルーチンは手続きのカプセル化だけ
  - データのカプセル化が十分でない
    - データを不必要に外部に晒してしまう



メッセージ

- オブジェクトにメッセージを送るのは、サブルーチンにパラメータを送るようなものですか？
  - 全く異なる概念
    - サブルーチン
      - 手続きにデータを送る
        - 相手がどんなデータ構造を扱えるか知っている必要がある
    - メソッド
      - データ（オブジェクト）に手続きを送る
        - 相手がどんな構造のデータか知る必要がない
        - 相手がどんなサービスを提供しているかは知らなければならない

多相と動的束縛

- 多くのメッセージに同じ名前を付けて混乱しませんか？
  - 異なる機能に同じ名前を付けると混乱する
  - 多相を使う方が概念が単純化する
    - 分析／設計／実現いずれ工程でも
- 動的束縛と静的束縛の違いは何ですか？
  - コンパイル時にメソッドが確定するのが「静的」
  - 実行時にメソッドが確定するのが「動的」
- 動的束縛の利点は何ですか？

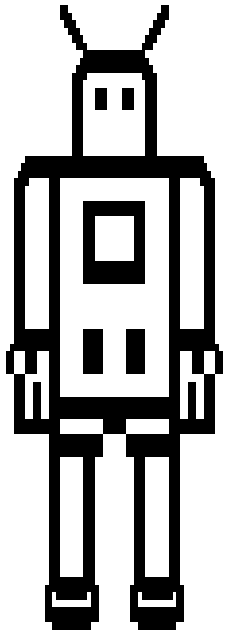
動的束縛の利点は何ですか？

```
if data = Text then
    displayText(data)
else if data = Picture then
    displayPicture(data)
end if
```

⇨

```
if data = Text then
    displayText(data)
else if data = Picture then
    displayPicture(data)
else if data = Image then
    displayImage(data)
end if
```

data display ⇨



⇨ Home

⇨

⇨

継承

- 継承を使った差分プログラミングと、サブルーチンを使った共通化の効果は同じくらいではありませんか？
  - 「再利用可能なモジュール構造の要件」が異なってくる
    - 型の変化に対応できる
    - データ構造とアルゴリズムの変化に対応できる
    - 関連した操作がまとまって定義されている
    - 顧客モジュールが実現方法を知ることなく操作を要求することができる
    - 実現方法の間の共通部分がうまくまとめられている

自動メモリ管理

- 自動メモリー管理をすると効率が悪くなりませんか？
  - 世代別ごみ集め（generation scavenging）法では、スーパープログラマー以外がメモリー管理するより速い
  - プログラマーがプログラミング時にメモリーの取得・解放をするのは、非常に難しい
    - 結果として、実行時エラーの嵐
      - 例
        - Windows, Windows NT
        - Mac OS
        - C, C++で構築されたほとんどすべてのシステム

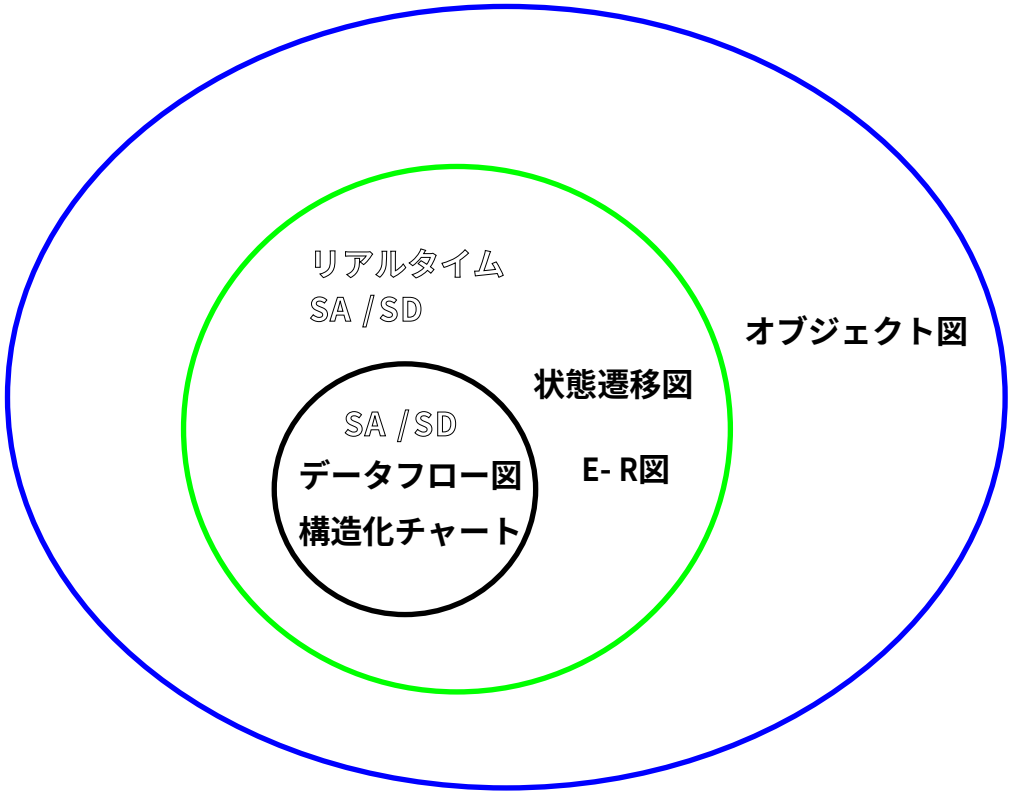
オブジェクト指向の位置付け

- オブジェクト指向言語とOODBMSの違いは何ですか？
- オブジェクト指向の位置づけ
- Unified Methodと他の手法の関係

オブジェクト指向言語とOODBMSの違いは何ですか？

- データの永続性
- オブジェクト指向言語
  - データの永続性の実現を目指す
    - OODBMSとの融合
- OODBMS
  - 照会言語の機能強化と標準化

オブジェクト指向の位置づけ



Unified Methodと他の手法の関係