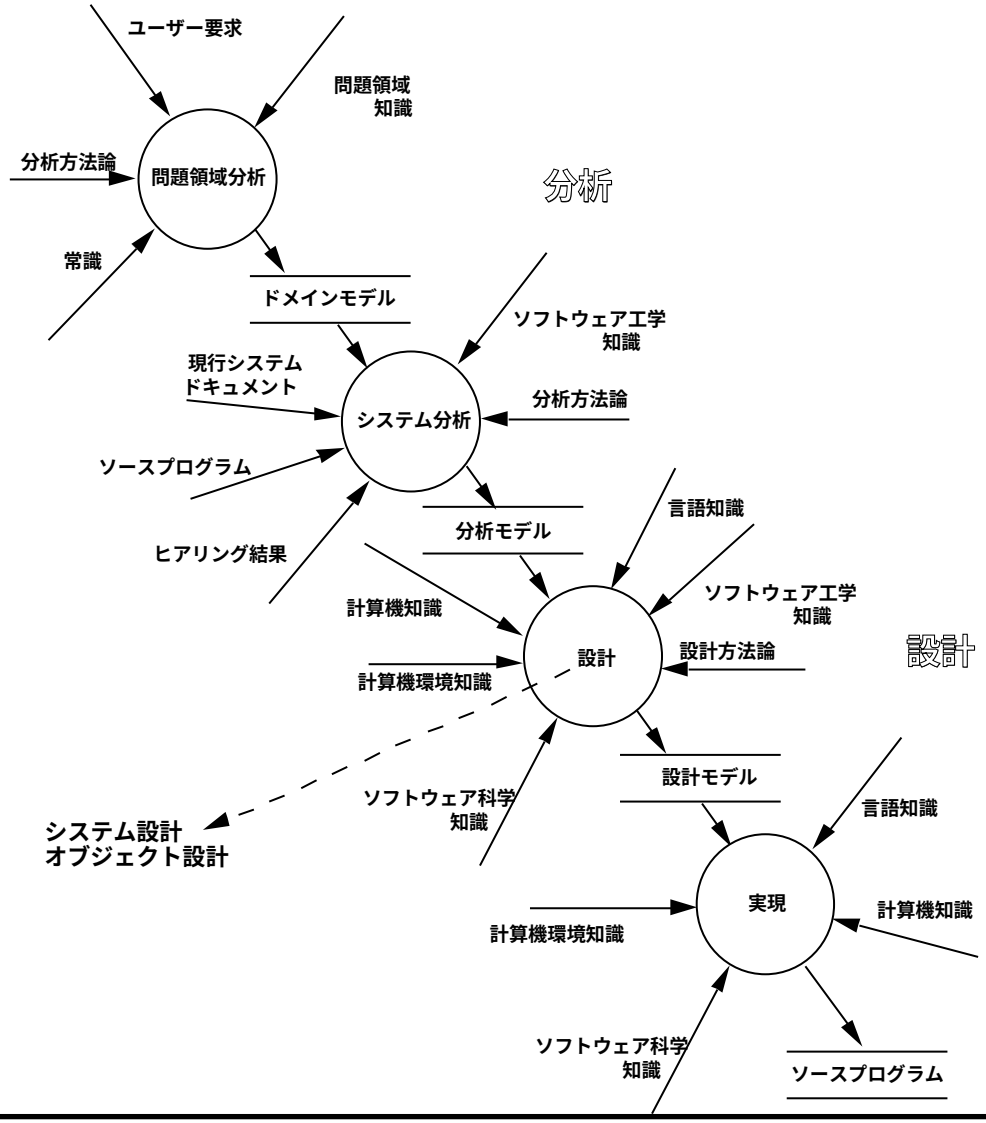
オブジェクト指向分析

- 手順は？
- 分析や設計局面の終了条件が見えない
- 誰が（あるいは、どれが）正しいオブジェクトと検証するのか？
- 手順は絶対か
- 分析は必ずデータ（オブジェクト）中心でやならければならないのか？
- ドキュメント化

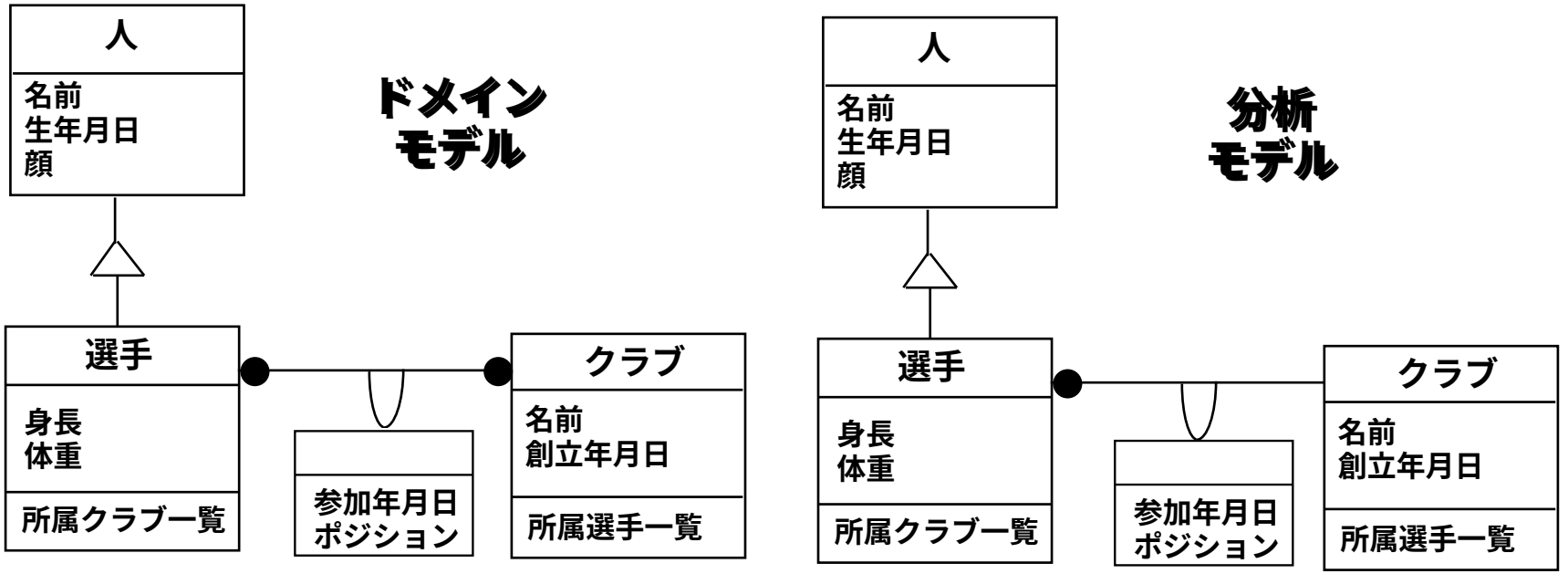
手順は？

- 分析／設計／実現の手順（概観）
- 分析の手順（イメージ）
- 設計/実現の手順（イメージ）
- 操作仕様記述のイメージ

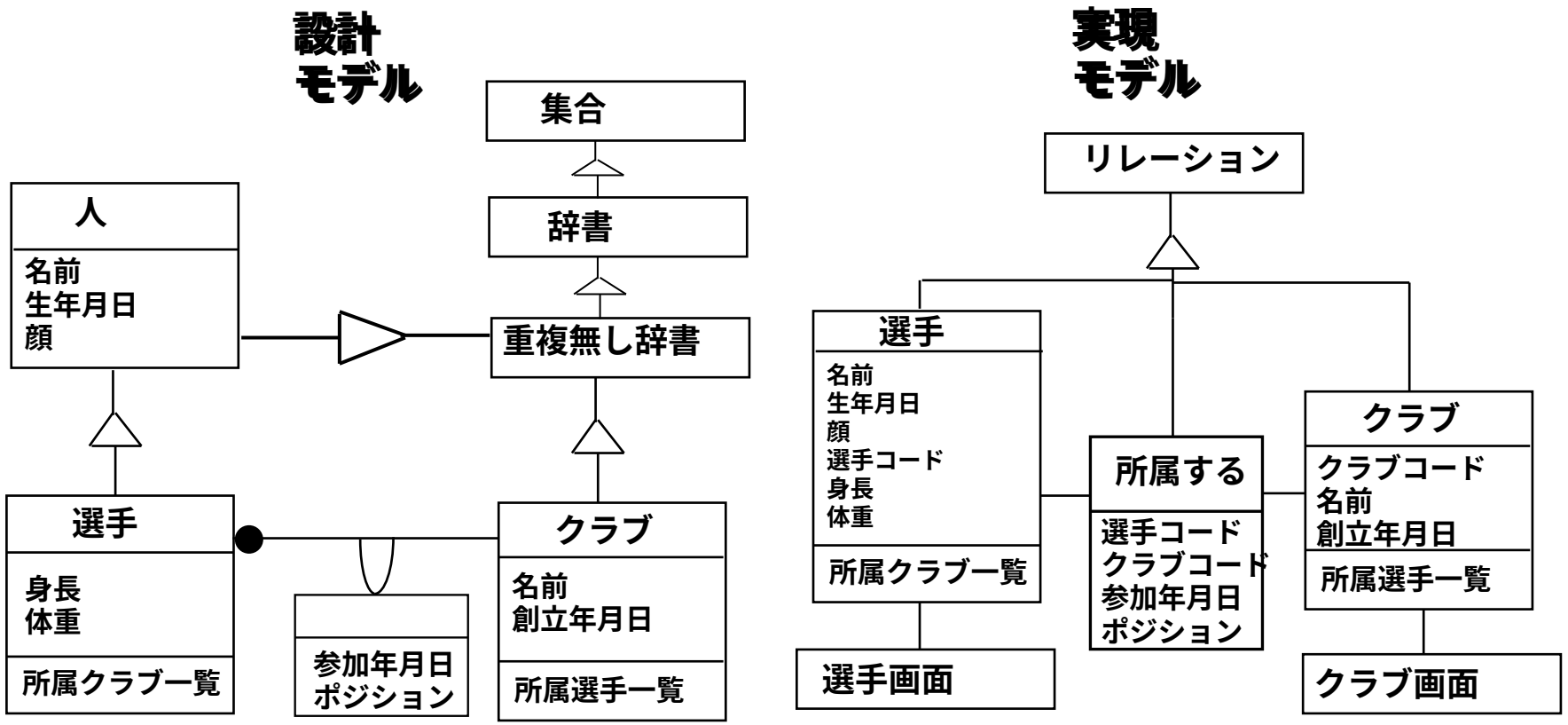
分析／設計／実現の手順（概観）



分析の手順（イメージ）


人
名前
生年月日
顔
ドメイン
モデル
選手
身長
体重
所属クラブ一覧
参加年月日
ポジション
クラブ
名前
創立年月日
所属選手一覧
人
名前
生年月日
顔
分析
モデル
選手
身長
体重
所属クラブ一覧
参加年月日
ポジション
クラブ
名前
創立年月日
所属選手一覧

設計/実現の手順（イメージ）



操作仕様記述のイメージ

「２つの日付の間の、指定された曜日の回数を求める」

pre
type R = {|rng [n → n / 7 | n∈Int]}
　/* 7で割った商の集合 */
f, t∈Int, w∈R, 0≦f≦t,
h: Int → R /* 環準同型（ring homomorphism） */

post
S = dom h(w) ∩ {f..t}・A ≡ card(S)
　/* Aが答え （dom h(w)≡h^{-1}(w)） */

分析や設計局面の終了条件が見えない

- シナリオに基づいたウォークスルーでエラーがなくなったら
  - 作成者が召集
  - 具体的なデータに基づいてテストする
  - その場で直さない
  - 誰の責任か問わない
  - マネージャーを入れない
  - １時間半程度で終える
- 再利用性・保守性分析をどこまでやるかは、プロジェクト方針次第
  - 一般には品質をできるだけ追求した方が、生産性が上がる
    - DeMarcoの法則（ピープルウェア）

誰が（あるいは、どれが）正しいオブジェクトと検証するのか？

- シナリオに基づいたウォークスルーで検証する
- 正しいモデルはいくつもある
- 間違いを徐々に排除していく
  - 主要目標
    - 保守性
    - 再利用性
    - 拡張性
  - 評価項目
    - 小さいモジュール
    - 単純なインタフェース
      - 少なく・小さく・明示的なインタフェース
    - 情報隠蔽
    - 解放／閉鎖の両立

手順は絶対か

- 手を抜いてはいけない
  - 「モデルを作るのが大変だから」というのは駄目
- エラーの検出効率を最大にするのが目的
  - この目的を逸脱しないなら、標準手順を変えてよい場合もある
  - 最初のうちは定石通りに
- オブジェクトモデルと動的モデルと機能モデルのあいだは何回もフィードバックする

分析は必ずデータ（オブジェクト）中心でやならければならないのか？

- データが重要で余り変化しないシステム（事務処理など）ではデータ（オブジェクト）中心
  - オブジェクトモデルを先に作る
    - データ（オブジェクト）の洗い出しが最初のステップ
- イベントが重要で、重要なデータがあまり無いシステムでは、振る舞い（イベント）中心
  - 動的モデルを先に作る
    - シナリオ作りが最初のステップ

ドキュメント化

- どの局面で「何」を実施し、
  どういう成果物を残すのかが見えない
- 成果物を作成するための具体的な作業項目が見えない
  - Booch法・Coad法などでは曖昧だが、UMLではっきりしている

どの局面で「何」を実施し、どういう成果物を残すのかが見えない

- Booch法などでは曖昧だが、UMLでははっきりしている
- 成果物
  - OOモデル
    - オブジェクト図
      - 操作の仕様を含む
    - 状態遷移図
      - シナリオ
    - OOデータフロー図
      - 操作の仕様を含む
      - オブジェクト・メッセージ図
    - シナリオ
      - Use Case
  - データ辞書
  - なぜそうしたかの記述
  - プログラム
  - プロジェクトの履歴