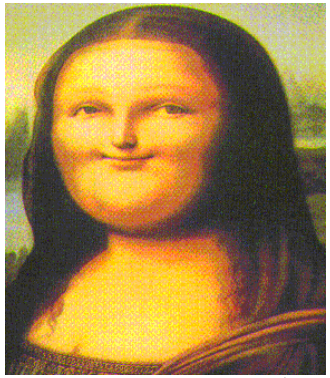
オブジェクト指向方法論

- 幻想と現実
- 方法論の選択
- 上流ＣＡＳＥの選択

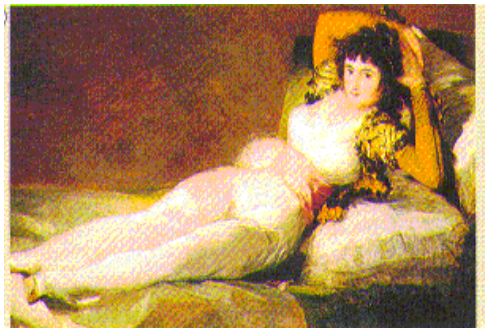
幻想と現実

- **「上流から下流まで一貫した自動ＣＡＳＥ」ができれば、ソフトウェア開発は不要になるのでしょうか？**
  - CASEの幻想
- コンポーネントウェアなどを使えば、分析／設計はあまり必要ないのではないですか？

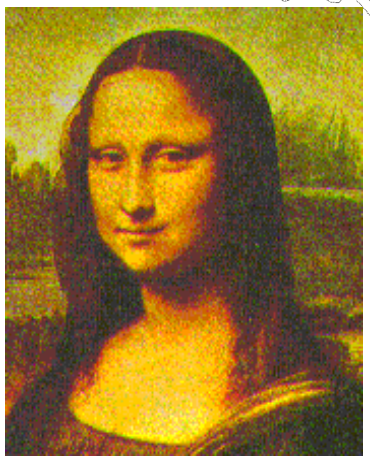
要求モデル



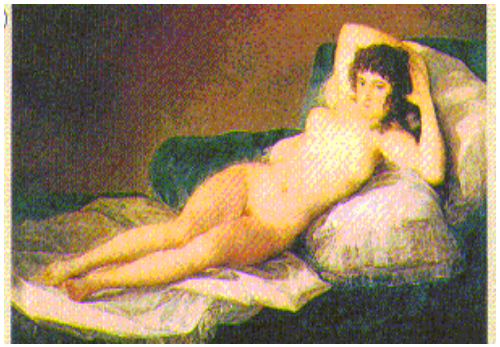
設計仕様



CASEの幻想



- 「ソフトウェア工学で分かったところを」というのが当面の目標



ユーザー要求

プログラム

コンポーネントウェアなどを使えば、分析／設計はあまり必要ないのではないですか？

- 何を作るかが分からなければ、ユーザー要求にあった物はできない
  - 何を作るのかを明確にするのが「分析」
- 正しい設計をしないと、保守性と再利用性が下がる
  - どうやって作るかを明確にするのが「設計」

方法論の選択

Unified Method Languageで標準化か？

- 方法論を選定する基準
  - 仕様記述言語が明確に定義されているか？
    - 宣言的に書けるか？
    - 「実行できない仕様」を、明確に書けるか？
  - 状態遷移図・ペトリネットなどの曖昧さのない動的モデルを使っているか？
  - オブジェクトモデルと機能モデルの対応が明確か？
  - 分析モデルと設計モデルで、統一した記法が使えるか？

上流ＣＡＳＥの選択

- どのCASEを使えばよいか？
- CASEを選定する基準は？

どのCASEを使えばよいか？

どのCASEを使えばよいか？

ツール名	記法	開発販売元	ＯＳ	日本語対応
Object Cast	OOA/OOD	FXIS	Mac, UNIX, WIndows	○
Objectory	OOSE	Objectory Corp.	Windows, UNIX	×
ObjectTeam, Paradigm Plus/Cadre	OMT, OOSA	Cadre Technologies	UNIX	△
ObjectTool	OOA/OOD	Object International	Mac, Windows, OS/2, UNIX	△
Paradigm Plus	OMT, OOSA, OOA/OOD,	Protosoft	Windows, WIndows NT,	×
Rational Rose	Booch, OMT	Rational, オージス総研	Windows, OS/2, UNIX	○
StP/OMT	OMT, OOSD	IDE, ニチメンデータシステム	UNIX	○
Turbo CASE	Booch	Struct Soft	Mac	×

CASEを選定する基準は？

- オブジェクト指向CASEの持つべき機能
  - 図エディター
  - ユーザインタフェース
  - 仕様記述言語
  - エラーチェック
  - ハイパーテキスト
  - 版管理
  - 構成管理
  - 手法のガイド
  - ブラウザー
  - データ辞書
  - 共有

仕様記述言語（RAISE, Z, OBJ, VDM）

集合

∈⊆⊇⊂⊃∪∩

論理

命題論理

∧∨¬⇒⇔

述語論理

∀∃

代数

```
value
  floor : Real → Nat      /* floor(1.3) ≡ 1 */
  sq_root : Real → Real   /* sq_root(25.0) ≡
5.0 */
  fl_sq : (Real → Real) → Real → Nat
        /* fl_sq(29.0) ≡ 5 */
axiom
[floor ]
  ∀r : Real, ∃ i : Nat・floor(r) as i
   post i ≦ r < i+1
[sq_root ]
  ∀n : Nat, ∃ s : Real・ sq_root(n) as s
  post s↑2 = n ∧ s ≧ 0.0
[fl_sq ]
  fl_sq  ≡ floor ° sq_root
```