

2004年度 新人研修資料
オブジェクト指向開発って何なんだ

2004-05-12

kdmsnr

<http://capsctrl.que.jp/kdmsnr/>

Agenda

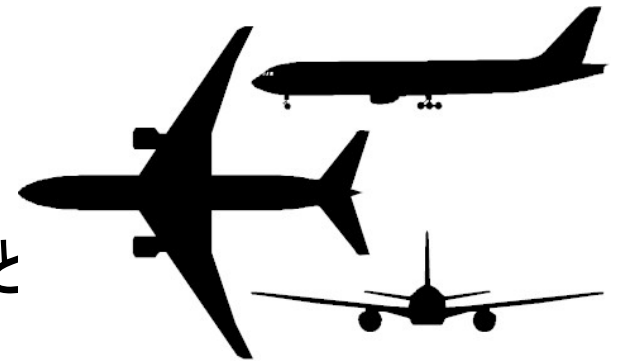
1. 「オブジェクト指向」って何？
2. 「オブジェクト指向型開発」って何？
3. それじゃあ、昔は何を使っていたの？
4. UMLダイアグラムいろいろ
5. UMLを利用するときの心構えとは？
6. 最近のトピックス

はじめる前に

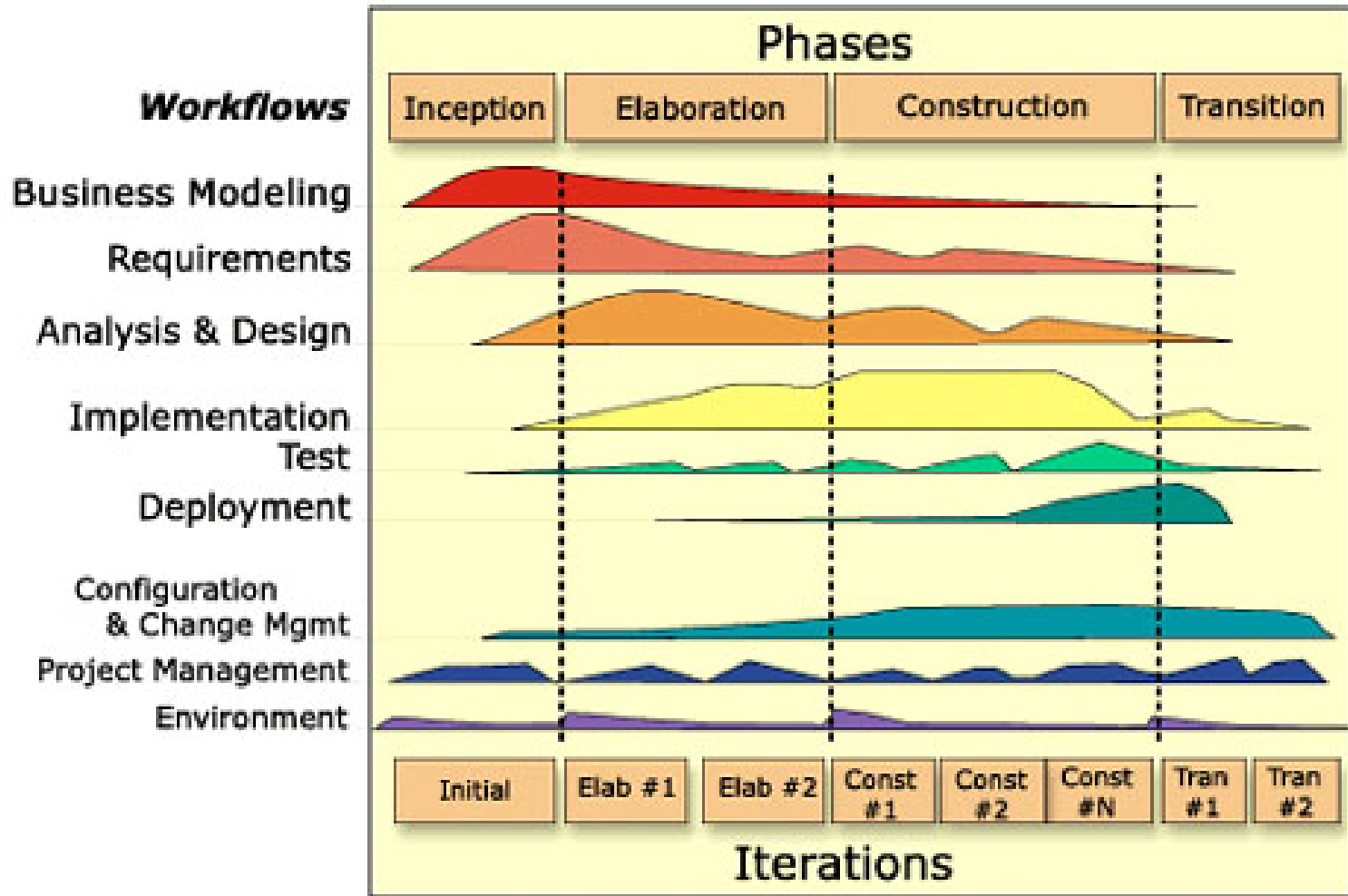
- 対象とするのは、「エンタープライズアプリケーション」
 - ワープロ、エレベーター制御、化学プラント制御、電話交換機、OS、コンパイラ、ゲーム、出会い系サイト... **じゃない!!**
 - 給与システム、患者記録、発送記録管理、原価分析、信用度採点、保険業務、サプライチェーン、会計、顧客サービス、外国為替取引... などなど、あー聞くだけでイヤになってくるね
- エンタープライズアプリケーションって
 - 技術的にはさほど高度じゃないけど
 - なぜかすごい複雑
 - なぜかすごいデータ
 - 企業と密接に結びついているから
 - いきなりシステム統合とか始まっちゃう可能性も
 - 古い技術を敢えて使わなきゃいけないかったりする
 - やり直しがものすごかったりする

0-1, 準備体操 – システムを作る前に

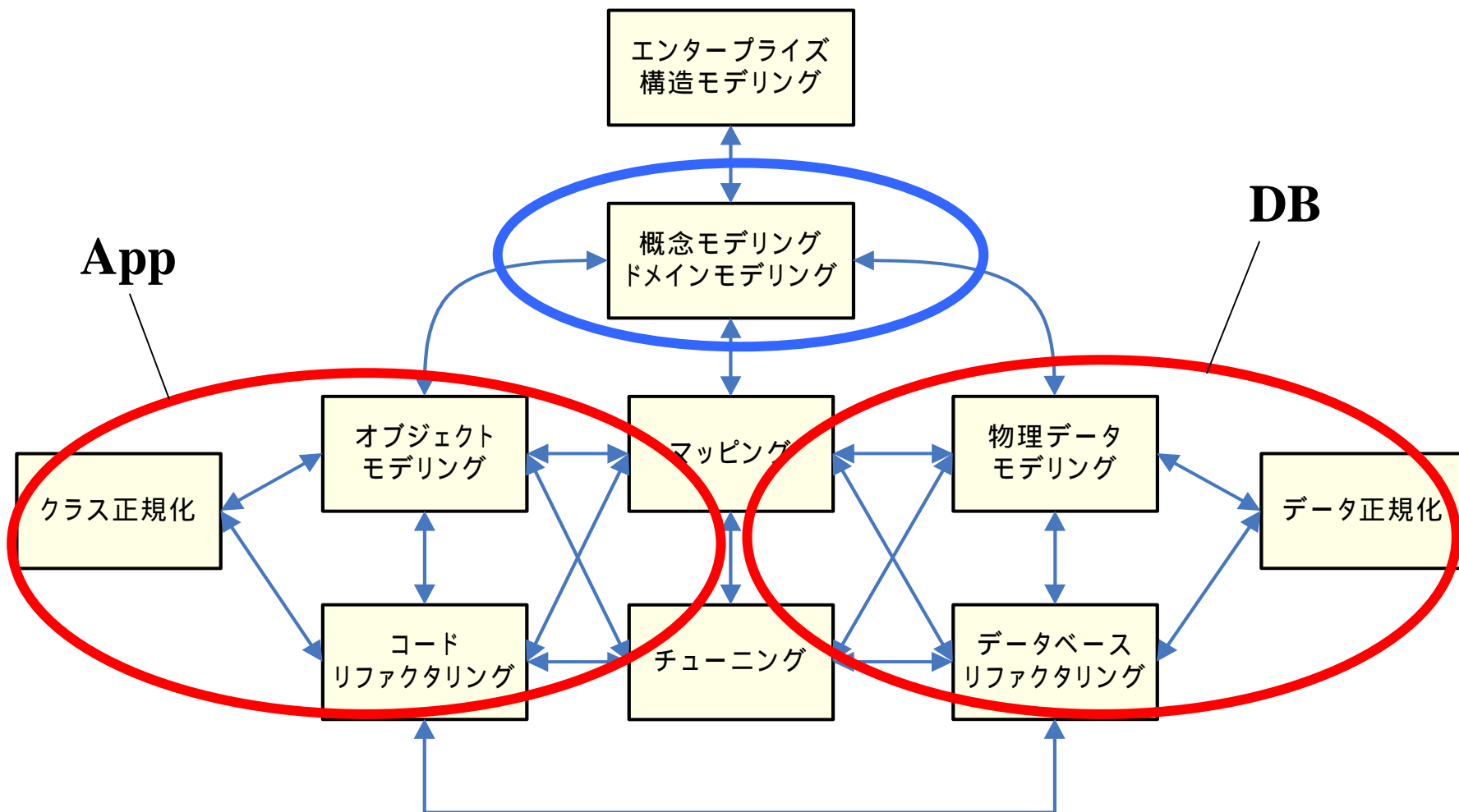
- 大きなものは手にもてるようにしよう
 - 大きなシステムをそのまま作ったりはしない
 - 小さくする？小さく切る？抽象化してみる？
- 抽象化
 - ある側面に注目する
 - 詳細にとらわれずに視野を広くする
- モデリング
 - 抽象化を使ってシステムを表現すること
 - 通常、複数のモデルを構築する
 - 各モデルは同じシステムを別の角度から見たもの



0-2, 準備体操 – システムを作るプロセス



0-3, 準備体操 – AppとDB



0-4, 準備体操 - 覚えたこと

1. システム開発で重要なのは？

- いかにしてシステムを扱いやすくするか

2. 開発の進め方はいろいろある

- Waterfall (ウォーターフォール型)
- iterative and incremental (反復的漸進型)
- Agile (アジャイル型)

3. Appの世界とDBの世界がある

- 両方とも同じものから派生している
- 別々の道を通るけど、最後に合体しなきゃいけない
- だけど、相容れないものがある
- DBというか、本当はデータの永続化と言ったほうが正しいかも
 - データの保存場所はファイルでもいいし、他のサービスでもいい。なにもRDBに限らない。だけど、RDBMSを使っているところが多いし、これからもRDBMSは使われつづけるだろう。ということで、とりあえずRDBMS。

1-1, 「オブジェクト指向」って何？

- システムはオブジェクトの集まりであるとする教え。
データと機能を合体させ「オブジェクト」を誕生させたのが始まり。WebとJavaの登場によって近年スターダムの階段も登り始めるも、日本ではデータ指向の古参者たちによって迫害されることも多い。
- オブジェクトって何だ？
 - アイデンティティがある
 - 自分の役割を知っているので「多態性」がある (polymorphism)
- オブジェクトを使うメリットって何だ？
 - 「カプセル化」を使った情報隠蔽が可能 (encapsulation)
 - 「継承」を使った差分プログラミングが可能 (inheritance)
 - オブジェクト指向という考えは(基本的に)一緒!!
 - Java, Ruby, C#, C++, Smalltalk, Python...
- もちろんデメリットもある。
 - 認知の世界っぽいところがあって難しい。スキルの差が出やすい。
 - RDBMSと合わない。

1-2, 「オブジェクト指向」って何？ (例)

Pet



age
height
weight
color

eat()
sleep()
say()

```
public class Pet {  
    private int age;  
    private float weight;  
    private float height;  
    private String color;  
  
    // public Pet...  
  
    public void sleep(){  
        String response  
            = "Good night. I will be "  
            + (height + 10) + "cm";  
        System.out.println(response);  
    }  
  
    public void eat(){  
        String response  
            = "Now " + weight + " kg: "  
            + "I'm so hungry!";  
        System.out.println(response);  
    }  
  
    public String say(String aWord){  
        String petResponse  
            = "What is " + aWord + "? "  
            + "I'm " + color + ".";  
        return petResponse;  
    }  
}
```

2, 「オブジェクト指向型開発」って何？

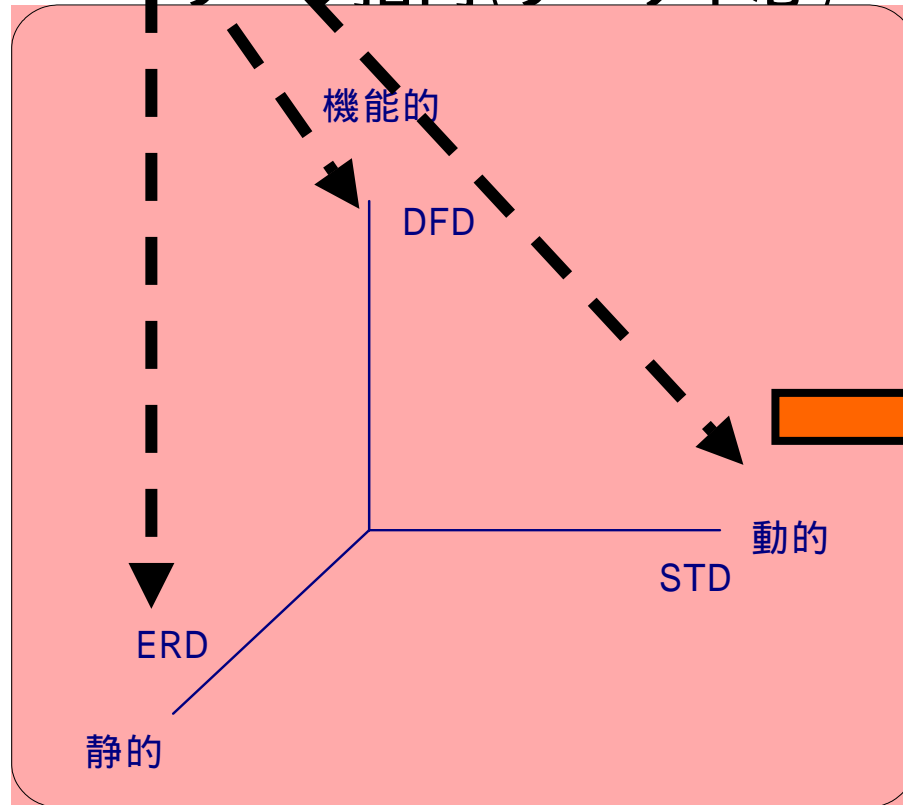
- OMT (James Rambough)
- Booch (Grady Booch)
- OOSE (Ivar Jacobson)



- Rational Unified Process + UML
 - RUP ... Rational (IBM)
 - UML ... OMG
 - v1.4, v1.5, v2.0

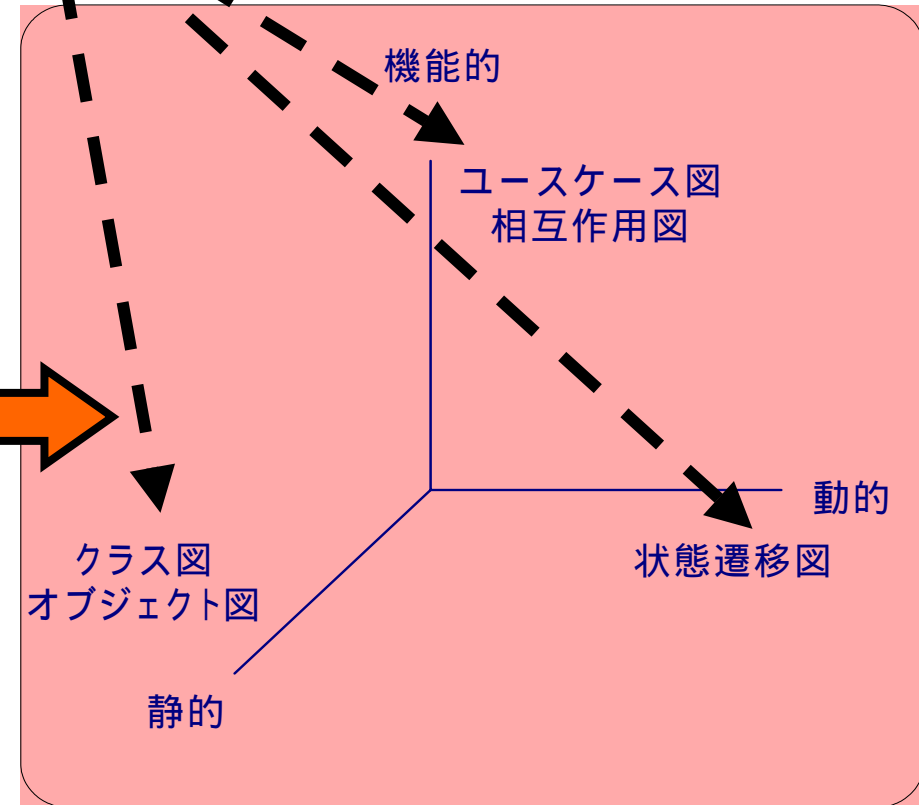
3, それじゃあ、昔(今も?)は何を使っていたの?

- GO-TO地獄時代
- 構造化(機能分化)
- データ指向(データ中心)



組み合わせが必要

- オブジェクト指向+UML



ひとつの考えと表記法でOK

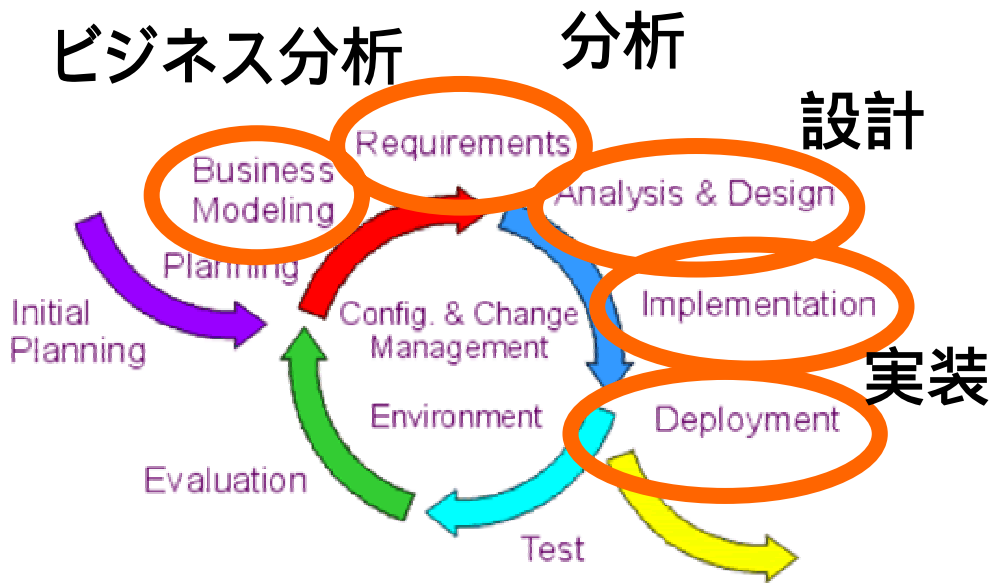
4, UMLダイアグラムいろいろ

1. ユースケース図
2. クラス図
3. コラボレーション図
4. シーケンス図
5. 状態図
6. アクティビティ図
7. 配置図
8. コンポーネント図

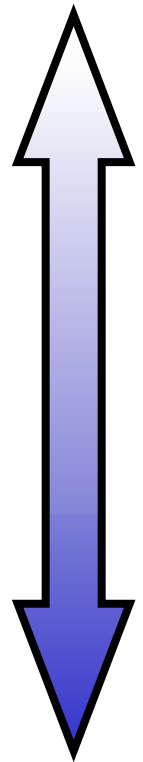
5, UMLを利用するときの心構えとは？

UMLを使うステージ

UMLを使うレベル



1. スケッチレベル
 - アイデアの整理
 - コミュニケーション
2. 設計図レベル
 - 詳細なモデル
 - コードの生成
3. MDAレベル
 - そのまま動く
 - 最先端(まだ未定)



6, 最近のトピックス

- UML2.0
 - より詳細な表記法(おれが書くのかー?)
- MDA
 - 詳細なモデルから動くコードを生成(まじでー)
- OO と、あと何か(OOだけじゃ解決できない)
 - Dependency Injection (IoC)
 - AOP
 - RDBMS以外のDBMS
- ORM
 - オブジェクトで作られた世界とRDBMSの世界とのミスマッチ
- DOA+(?)
 - OOじゃ、やっぱりダメなの?(まさか)