

noUML



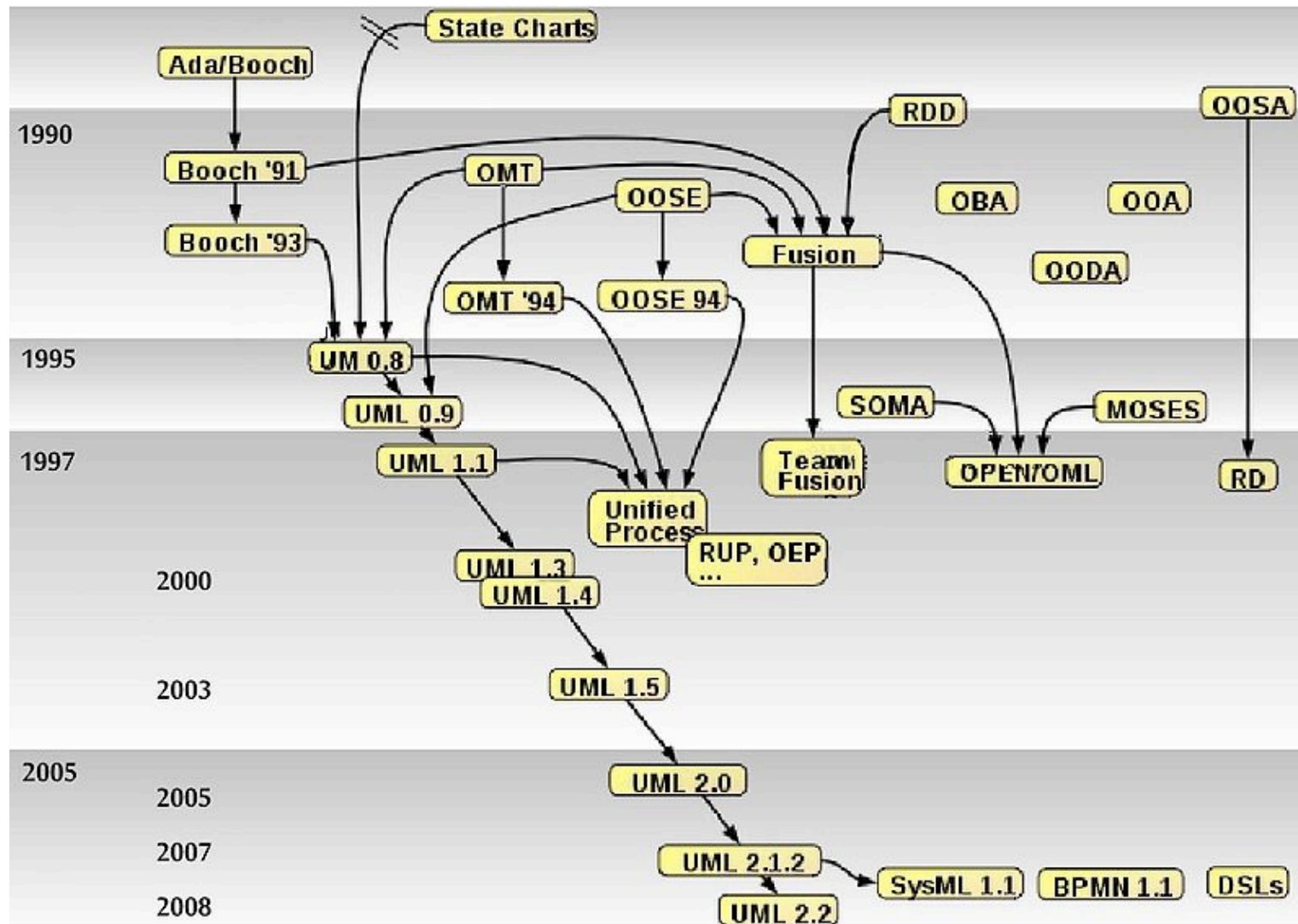
masaki@metabolics.co.jp

2010.07.24



- ✿ *Groovy*とも*Grails*ともあまり関係ないけど
- ✿ *UML*だめじゃん,なんとかしよう
- ✿ という話をします

- ◆ *Unified Method*と呼ばれていた0.8の頃
(1995)からずっとUMLをフォローしてきました
- ◆ 本も書いたし, お金もだいぶ稼がせてもらいました 😊



UML,
もちろん良いところも,
貢献してきた部分もあるんだけど,
このままで良いのか?

ダメなところ I

- ✿ *UML2*は複雑すぎる, *OMG*技術委員会のおもちゃになってる
- ✿ *UML2.0*策定中の2001年頃からそんな記事は書いてるんだけど
- ✿ 今日はそれには触れません

ダメなところ II

- ◆ そもそもUMLにダメなところがあった
- ◆ 最初はみんなそれを理解した上で使っていたけど、普及するにしたがって、UMLがオブジェクト指向やソフトウェア構築を縛る枠になってしまった
- ◆ 呪縛!

ソフトウェアエンジニアリングの呪縛



例えば

✦ “クラス”概念が中心にありすぎる!

✦ オブジェクトの本質はクラスじゃない

- ◆ オブジェクトの本質はオブジェクト間のメッセージ交換であって、継承でも関連でもない!

methodMissing()

- ◆ 継承はメッセージ交換パタンの一つ,
- ◆ 関連は (本来は動的に変わる) メッセージ送信先に過ぎない

- ◆ クラスは, オブジェクトとは異なるドメイン (オントロジ) から密輸入された概念である
- ◆ もちろん, これはこれで重要だけどね

Groovyプログラマにお馴染みのもの

- ✦ *delegate*
- ✦ *aspect*
- ✦ *dynamic injection of methods*
- ✦ *metaclass (per class, per instance)*
- ✦ *closure*
- ✦

- ◆ こういう *Groovy* プログラマになじみ深い、
必要不可欠な概念を *UML* でどうやって書
いたらいい?

考えられる反論

- ◆ UMLはプログラミング・レベルの概念を記述するためのものではない
- ◆ もっと上のレベルの概念を記述するためのものである

反論に対する反論

- ◆ 上のレベルって何だ?
- ◆ 設計だとしたら,むしろUMLが柔軟な設計の可能性を台無しにしている
- ◆ カビ臭い,変なアーキテクチャがはびこる要因

◆ 上のレベルって何だ?

◆ 仕様/要求/問題記述だとしたら, *UML*

だってまったく記述できていない!

◆ むりやり *UML* を拡張してなんとかしているだけ

◆ *DOA* っぽい, 変なデータ偏重主義の要因

noSQL と noUML

noSQL

- ◆ “データ構造は表や集合で表されるようなものばかりではないことに (やっと) 気付きました”
- ◆ 多様なデータ構造がある
 - ◆ *b-tree, key-value, network, parallelism, ...*

GroovyでもCollectionに対する操作と制御構造を混同している人は多いよね

◆ 今のUMLは表/集合に偏った見方を煽っている

クラス図を見てもER図にしか見えない人たちが...

◆ noUMLは多様なデータ構造も表せなければならない

さてどうしようか

UMLのいいところ I

- ◆ ビジュアルであるところ
- ◆ 見た目だけじゃなく, もっとインタラクティブに行きたいところだ (標準規格としてはしょうがないけど)

UMLのいいところ II

- ◆ モデル駆動開発のプラットフォームであるところ
- ◆ いまいちだけど:-)
- ◆ でも彼らが頑張ってくれたお陰でダメなところが見えてきてはいないか

✿ この二つの長所を残しつつ

✿ 本来のオブジェクト・モデリング

✿ 多様なデータ構造表現

✿ ができないか

マルチタッチとかでな:-)

ここから先は
これから考えます

presented to you by ...

meta  **bolics**