こんなに簡単EoD

ORACLE JDEVELOPER 10g

日本オラクル株式会社
マーケティング本部
担当ディレクター 西脇 資哲

本セッションは...

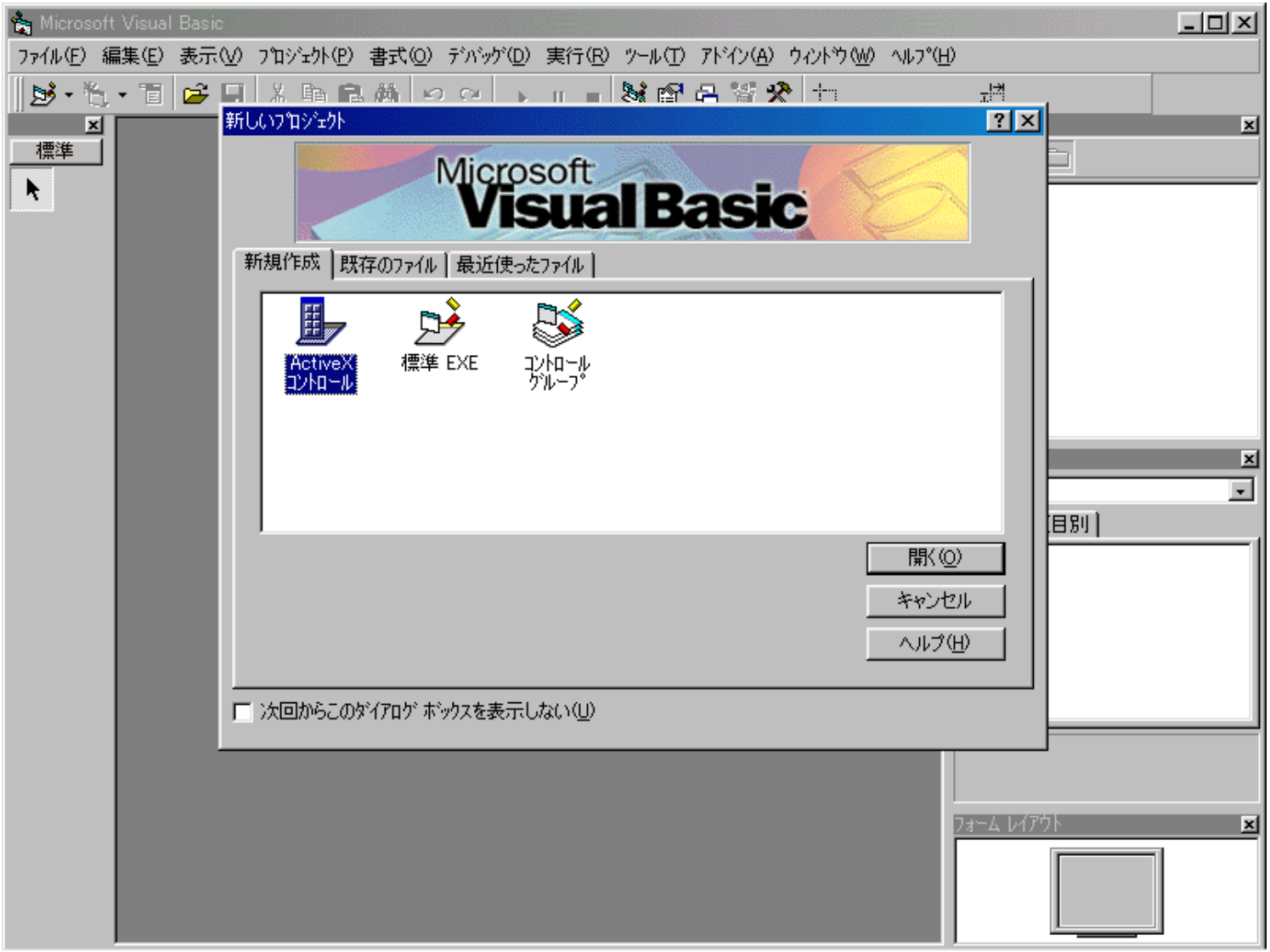
- 7月23日に発売開始され、ソフトウェア販売ランキング<プログラミング部門>のトップを走りつづけるJDeveloper 10gについて紹介します。
- 単純な製品紹介ではなく、魅力的な機能や生産性を向上させる手段にフォーカスをあてて解説を行います。

目次

- アプリケーションの生産性とは何か?
  - J2EEアプリケーションの開発工程と生産性
- ビジネストランザクションという管理概念
- まとめ

あの時は感動しました
(正直)

これです



だって、誰にでも マウスで
こんなアプリケーションができたから...

ところが…

感動しません…（正直）…なぜか…

ようやく Java にも EoD

- 複雑に高度になってきたJava
- ここでようやく EoD に振り向いてくれました
  - EJBの開発効率を大幅に軽減
  - 言語の仕様も改善
  - JSFでアプリケーションの開発手法（工程）も改善
- ツールも充実

真剣に生産性を考える

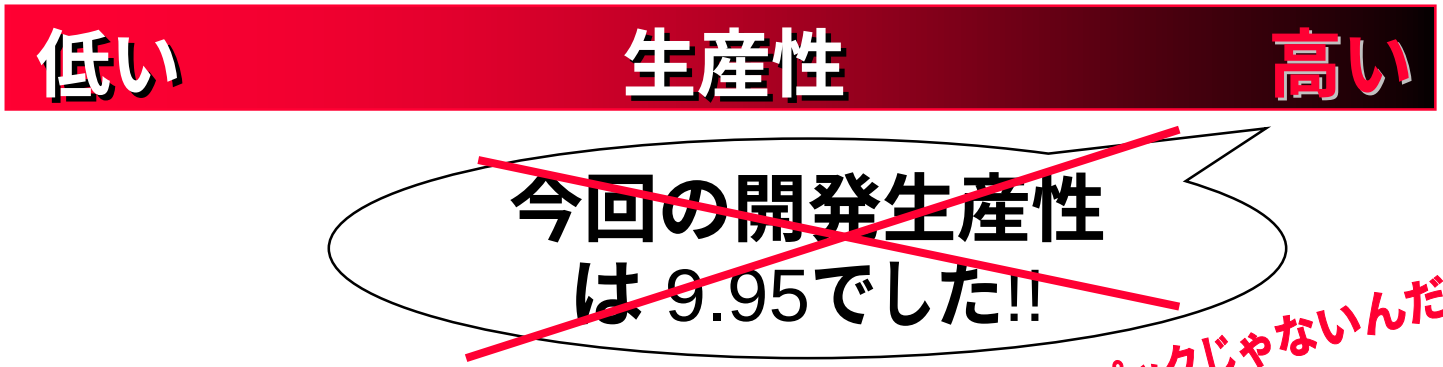
アプリケーションの生産性とは?

- 数値が高いほど効率がよい
- 数値が低いほど効率が悪い

と一般的に言われており、
開発プロジェクトの責任者が気にする指標

アプリケーションの生産性とは?

- でも、数値化された指標は見たことが無い



あるのは、
開発期間、開発工数、FP/行数/モジュール数などから
算出された値だけ?、感覚だけ?、売上と工数だけ?

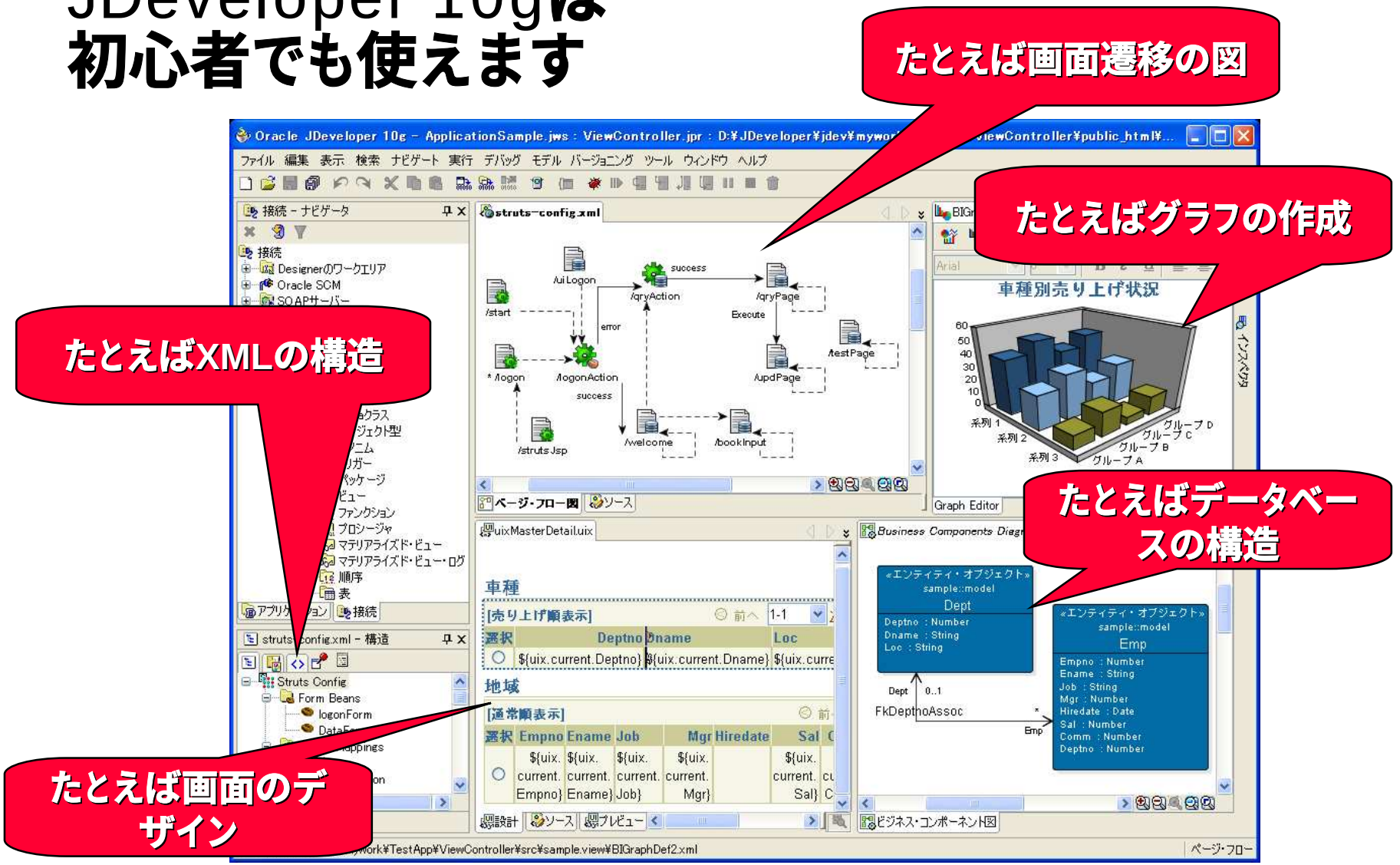
生産性が高いとはどういうことか?
たとえば、

- 期間: 短い期間で開発できる
- 工程: 開発工程が短縮できる
- コード: コーディング行数が少なくてすむ
- 品質: バグが少ない、バグが発見しやすい
- 保守: ドキュメントが自動的に生成される
- 他: 後戻り工程が少ない、後戻りが楽
- コスト: 開発環境が安い、教育コストが安い
- 性能: 性能が良いアプリケーションが作れる

期間：
短い期間で開発できる

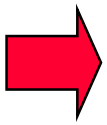
- 短い期間で開発するためにはどうすればよいのか?
  - エキスパートをそろえる
  - 初心者/中級者でも使えるツールを使う
  - チーム開発をする

JDeveloper 10gは
初心者でも使えます



知られていないチーム開発機能

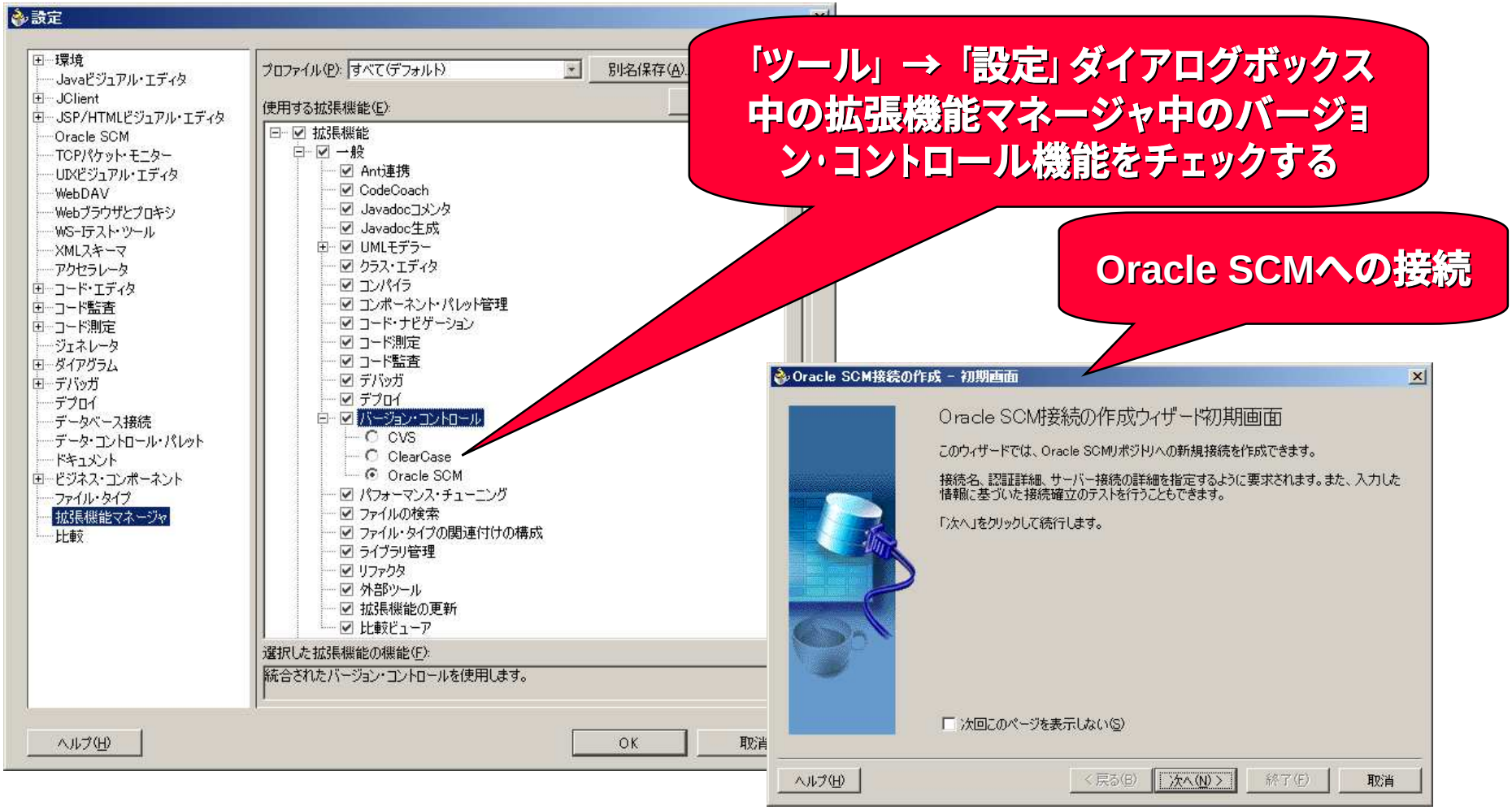
- Oracle Software Configuration Manager*
  - 強力なソースコントロール機能
    - チェックイン･チェックアウト
  - ソース変更の履歴管理、履歴ビューア
  - ソースの比較
  - バージョニング
  - ファイルレベルでのパッチの適用など



これらの機能を使うと複数メンバのプロジェクト進行は容易

＊旧Oracle Repository

チーム開発機能を有効にする



他に接続できるプロジェクト管理製品

- CVS
  - Concurrent Versions System
  - JDeveloper10gと同じマシンにインストール
  - CVSはhttp://www.cvshome.org より入手

工程:
開発工程が短縮できる

- 開発工程を短縮するにはどうすればよいか?
  - 開発工程をスキップする
  - 開発工程の一部を自動化する

開発工程のスキップ

	オブジェクト指向的	データ指向的	EUC的
設計 (UMLなど)	①設計		
設計 (クラス設計/DB設計)	②DBオブジェクト生成 (TOPLink的 O→R)	①DB設計	②DB設計
コーディング	③ビジネスロジック作成	②コード作成 (R→O ひな型作成)	③コード作成
UI画面のデザイン	④画面作成	③画面作成	①画面作成
デバッグ	⑤デバッグ	④デバッグ	④デバッグ
チューニング	⑥チューニング	⑤チューニング	⑤チューニング

開発工程の一部を自動化する

- 絵→コード（オブジェクト）の自動化は可能

例1）Business Component Diagramから
データベースオブジェクト

例2）Java Class DiagramからJavaオブジェクト

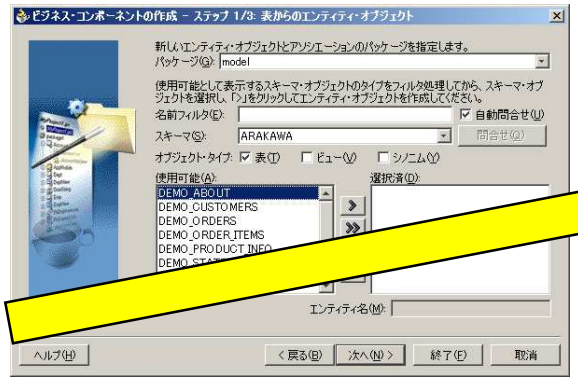
コード:
コーディング行数が少なくてすむ

- コーディング行数を少なくするにはどうすればよいか?
  - コードの自動生成機能を使う
  - フレームワークを使う
  - コーディングの物理的量を減らす
  - コーディングの手助けをする

豊富なコード自動生成
→いわゆるウィザード

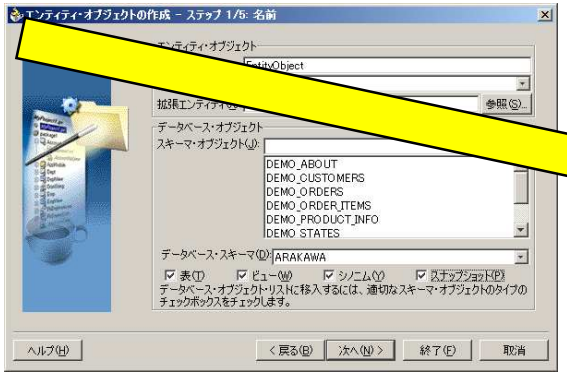
ビジネスコンポーネントの作成 ウィザード

JavaBeansから



データベースにアクセスする
Javaデータモジュール

データベースの構造から

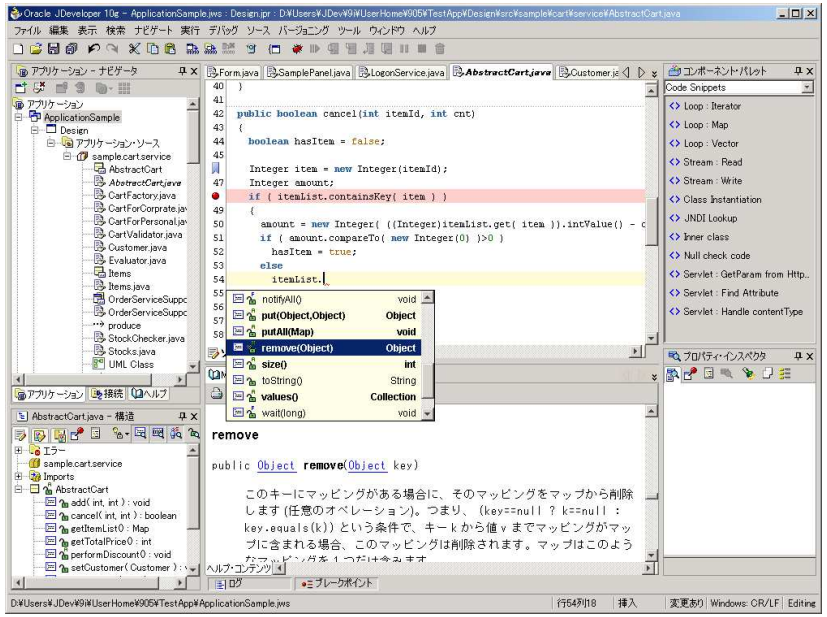


Entity Bean

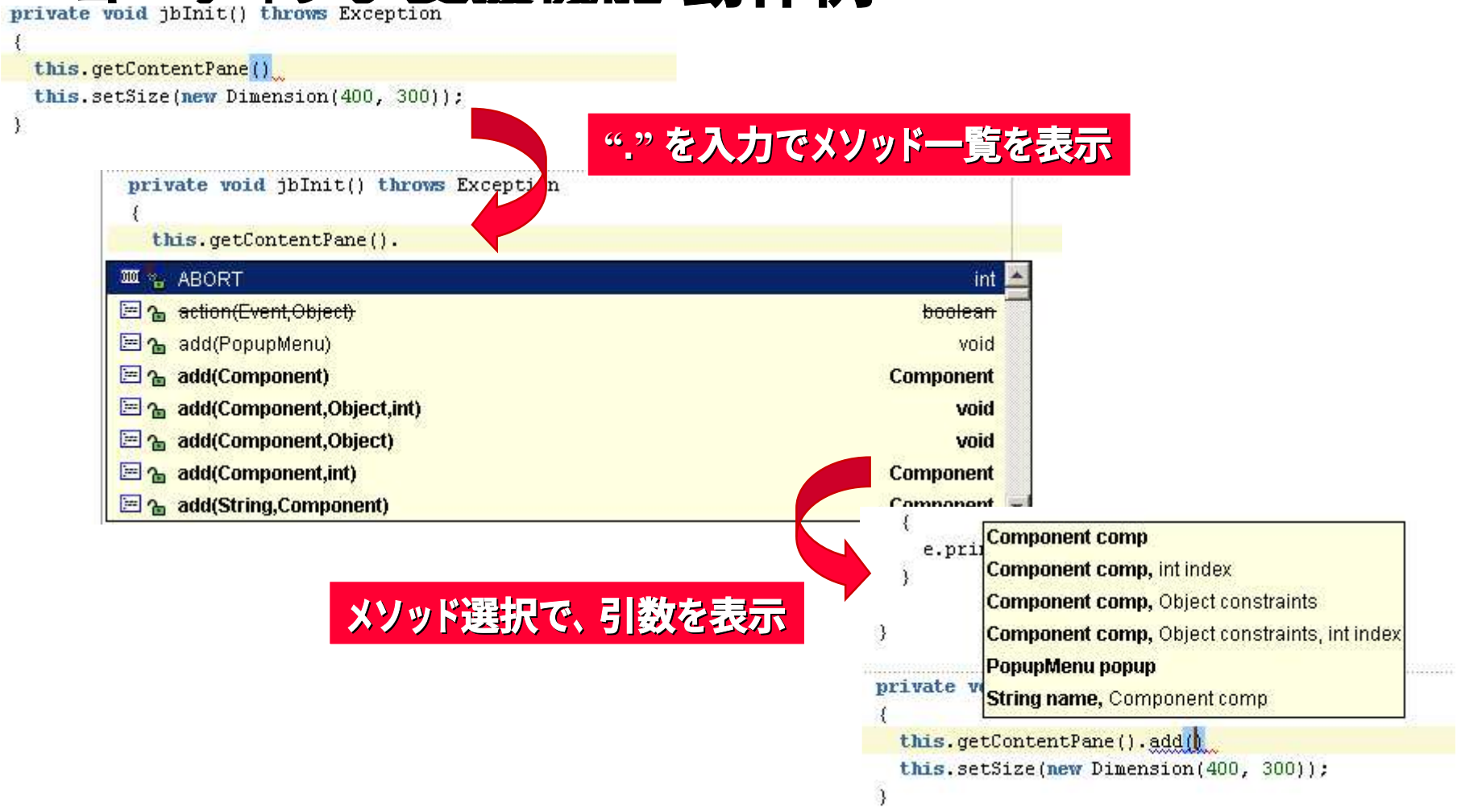
エンティティオブジェクトの作成 ウィザード

コーディングの物理的量を減らす

- コーディングを支援する機能を使う
  - 命令の途中で候補を表示
  - 引数/戻り値などを常にチェック
  - import文を自動追加
  - JavaDocをいつも参照



コーディング支援機能 動作例



Quick JavaDoc機能

- **いつでもカーソル位置の適切な** JavaDoc **を表示**

品質：
バグが少ない、バグが発見しやすい

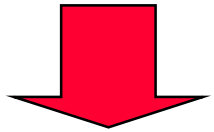
- バグを発見しやすくし、少なくするためには?
  - 行儀の良いコーディングをする
  - コーディングの無駄を無くす
  - 強力なデバッガを使用する

J2EE仕様の正しい理解

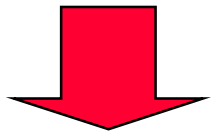
- J2SEのバージョン
  - どのバージョンをターゲットとしているのかを明確に
- J2EEのバージョンの違い
  - どのバージョンの仕様に基づいて開発するのか明確に
- J2EE仕様の誤差
  - 仕様には必ず実装時に誤差が生じる
- J2EEのオプション扱いの仕様の確認
- アプリケーション・サーバ固有の機能

行儀の良い、無駄の無いコードは?

コーディングで消し忘れた無駄なメソッド、変数
解放していないメモリ
非常に複雑な継承ツリー



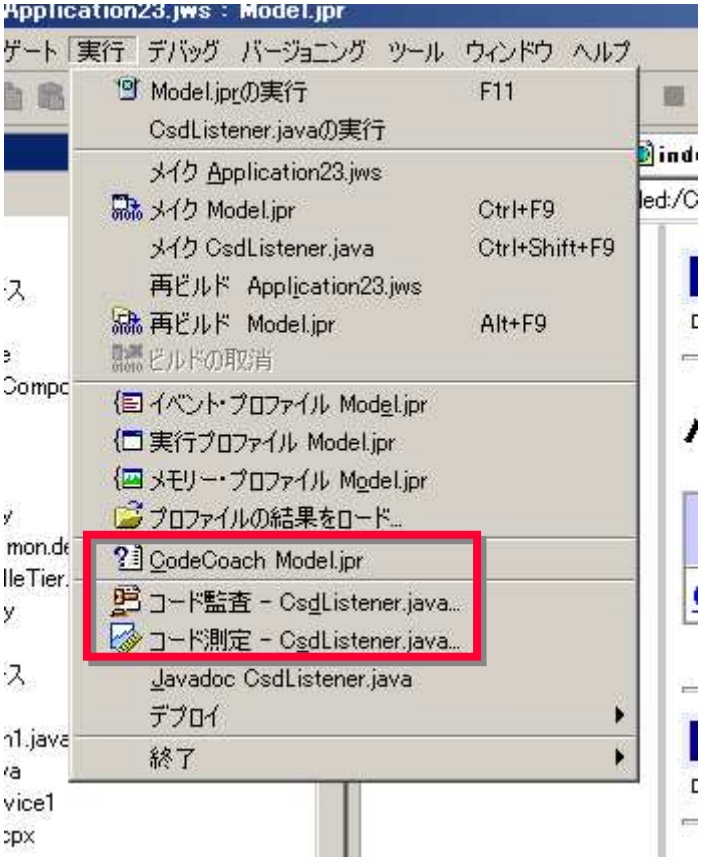
大規模な開発中に、気にしていられません



一貫した指標のもとに自動チェックしてくれる機能

その上で行儀の良いプログラムにするには?

JDeveloper10g のコード改善のための機能



Javaアプリケーションの
ソースコードをすべてチェック

- コーディング規則
- クラス、メソッド
- フィールド、変数の状況
- 継承ツリーの深さ
- 構成メンバのサイズ
- 循環的複雑度 (分岐複雑度)

デバッグのポイント

何をデバック するのか?

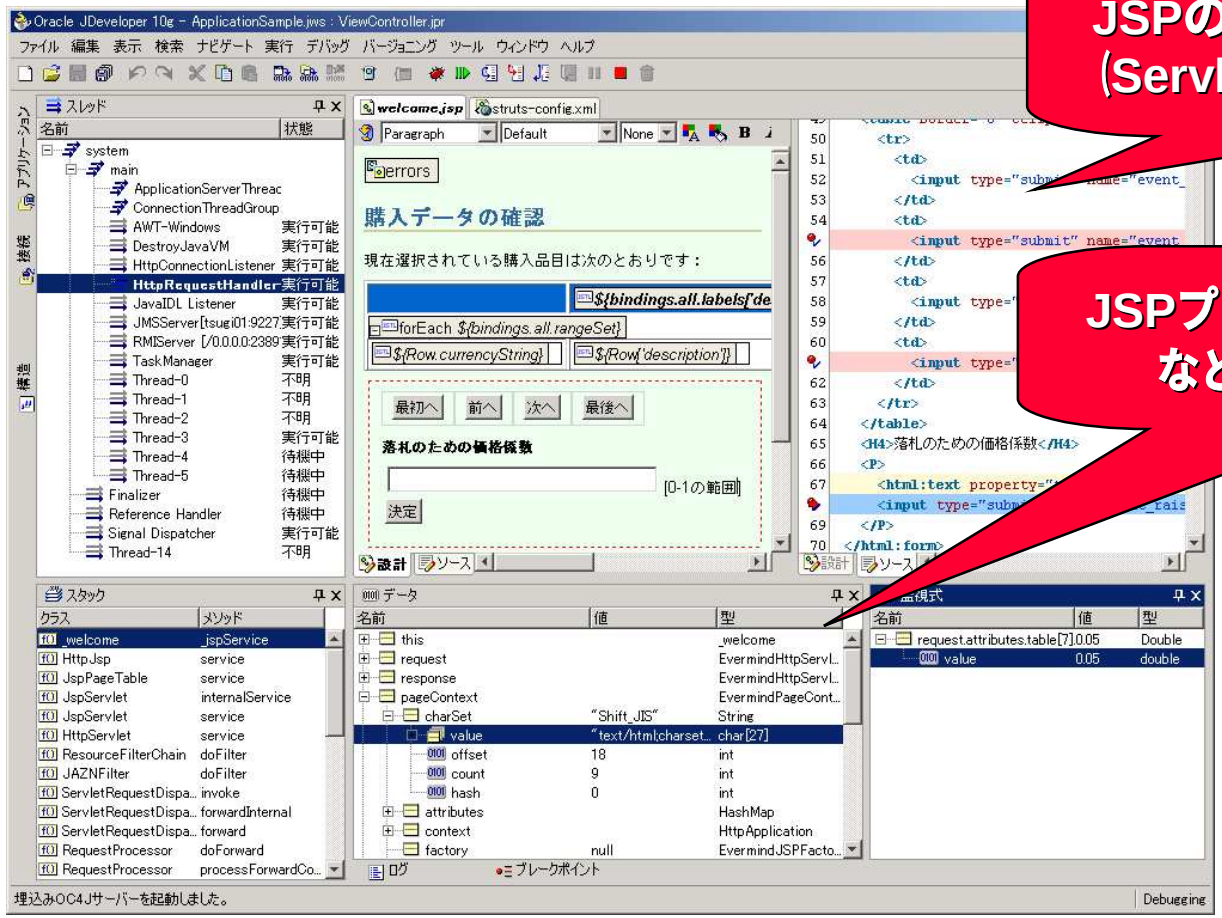
HTML?

JSP/Servlet
EJB…

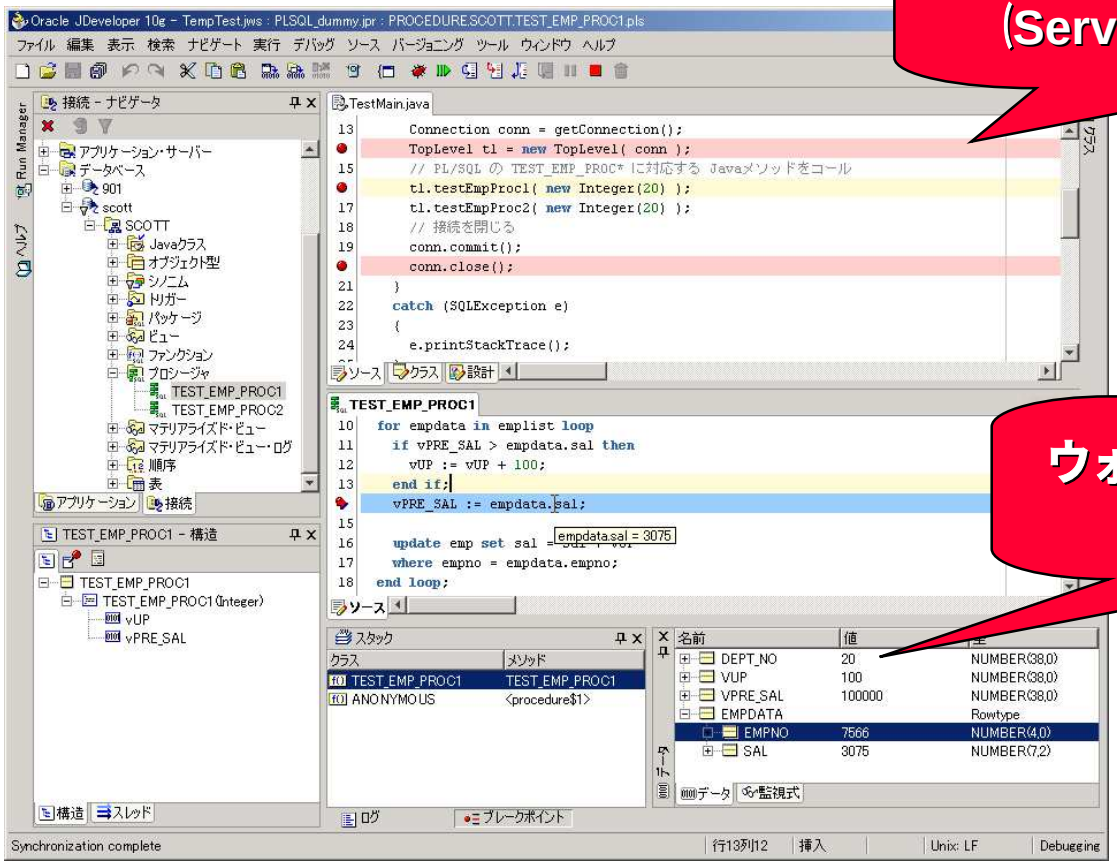
PL/SQL

どうやって デバック するのか?

例)
JSPのデバッグの様子



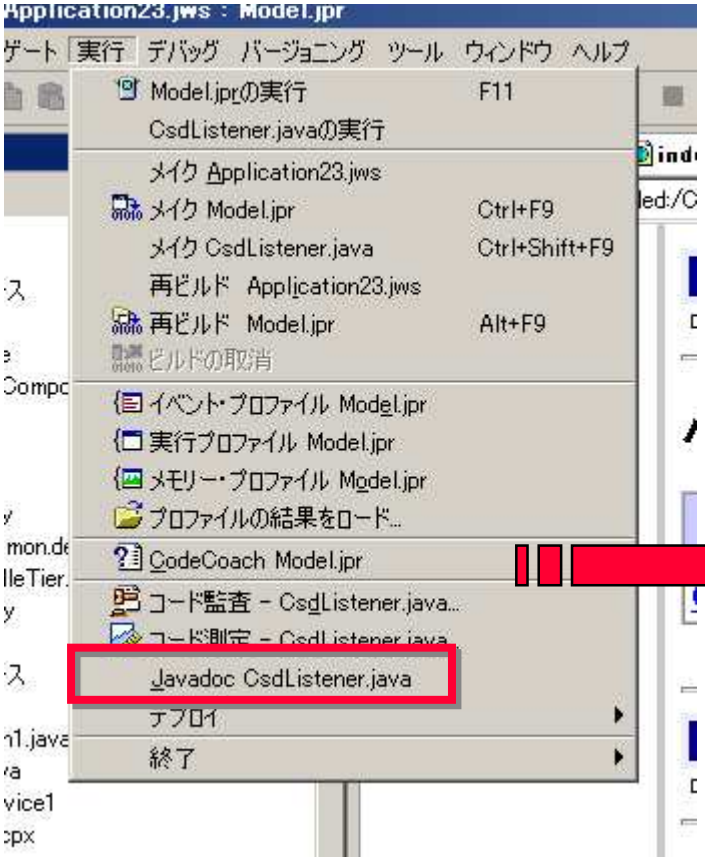
例)
PL/SQLデバッグの様子



保守：
保守に必要なドキュメントを生成

- 開発されたアプリケーションの保守性を高めるためのドキュメントはどうするべきか?
  - ドキュメント生成機能
  - 設計フェーズに戻る
  - 移植性の高いアプリケーションにする

ドキュメント作成方法 その1
Javadoc機能



自動生成された HTML

model

クラス CsdEvent

java.lang.Object
  └java.util.EventObject
    └**model.CsdEvent**

すべての実装インタフェース:
java.io.Serializable

public class **CsdEvent**
extends java.util.EventObject

関連項目:
直列化された形式

フィールドの概要

クラス java.util.EventObject から継承したフィールド

source

コンストラクタの概要

CsdEvent(java.lang.Object source)

クラス java.util.EventObject から継承したメソッド

getSource, toString

クラス java.lang.Object から継承したメソッド

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

ドキュメント作成方法 その2
コードから設計フェイズに戻る

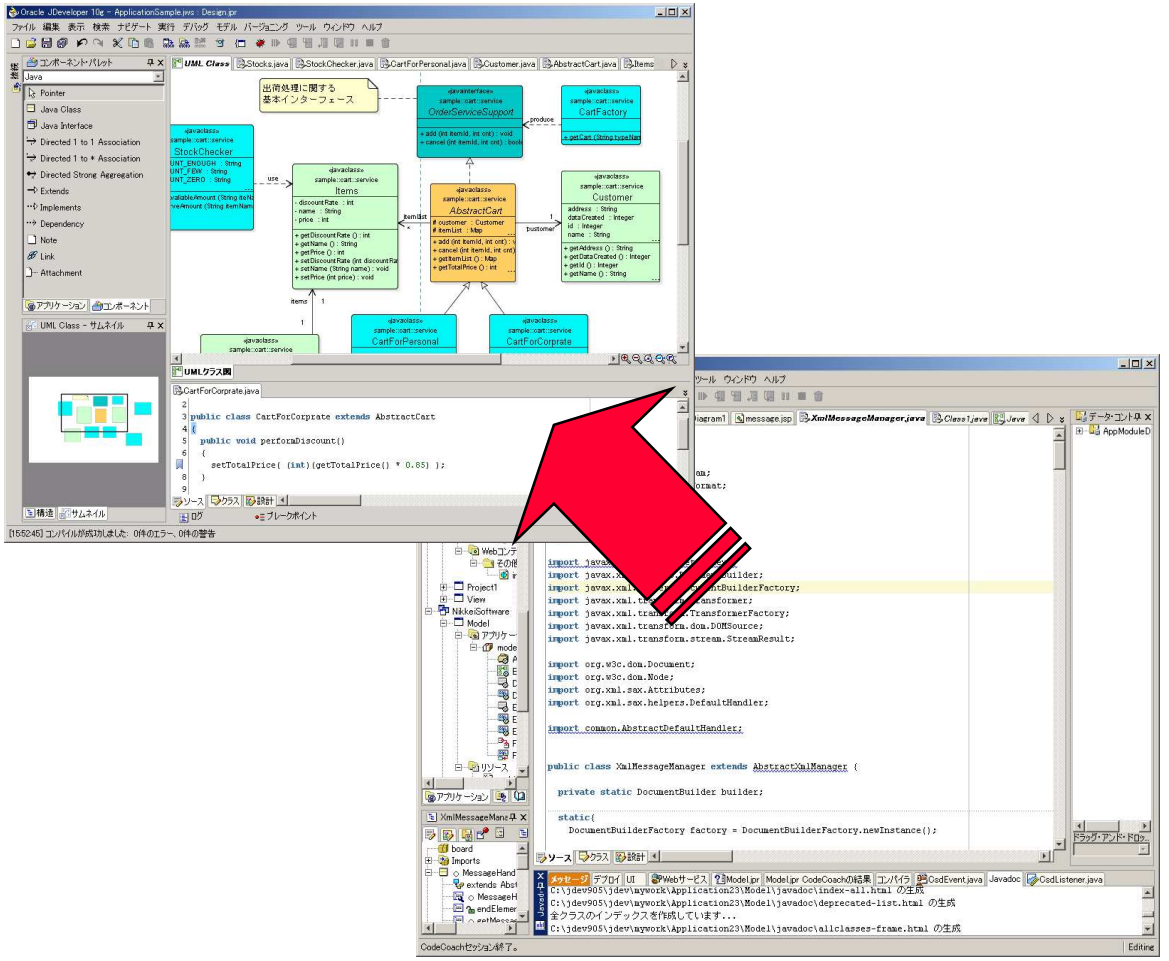
設計 (UMLなど)

リバース

設計 (クラス設計/DB設計)

リバース

コーディング



他:
後戻り工数が少ない、後戻りが楽

- 後戻り工数を減らし、楽な開発をすすめるためには?
  - フェーズを後戻りしても大丈夫な機能
  - どの工程からでも入れる機能

コスト:
開発環境が安い、教育コストが安い

- 開発環境、教育コストを安くするには?

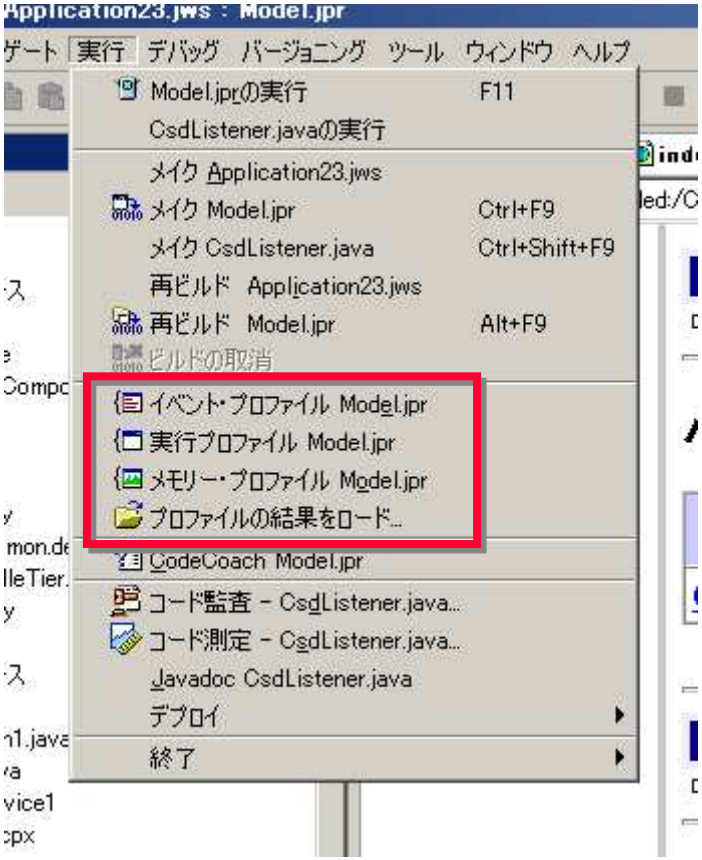
ソースネクストより
1,980円で発売中

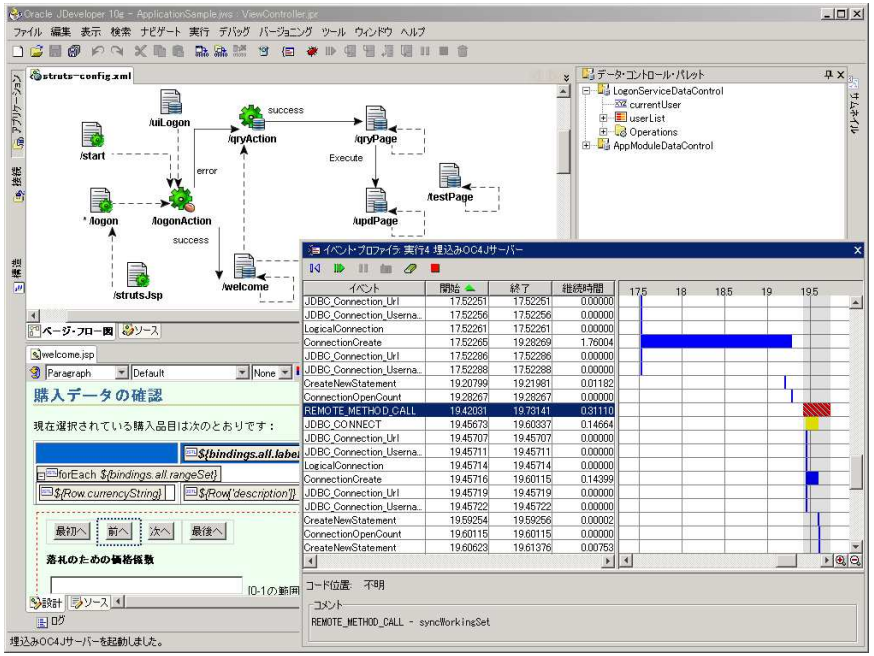
性能：
性能の良いアプリケーションが作れる

- 性能の良いアプリケーションを作るには?
  - パフォーマンスチューニング機能を使う

性能が良いかどうかを確認する方法

JDeveloper10g のプロファイリング機能





アプリケーションを監視下で実行

- メモリ使用状況
- 各メソッドの実行時間計測

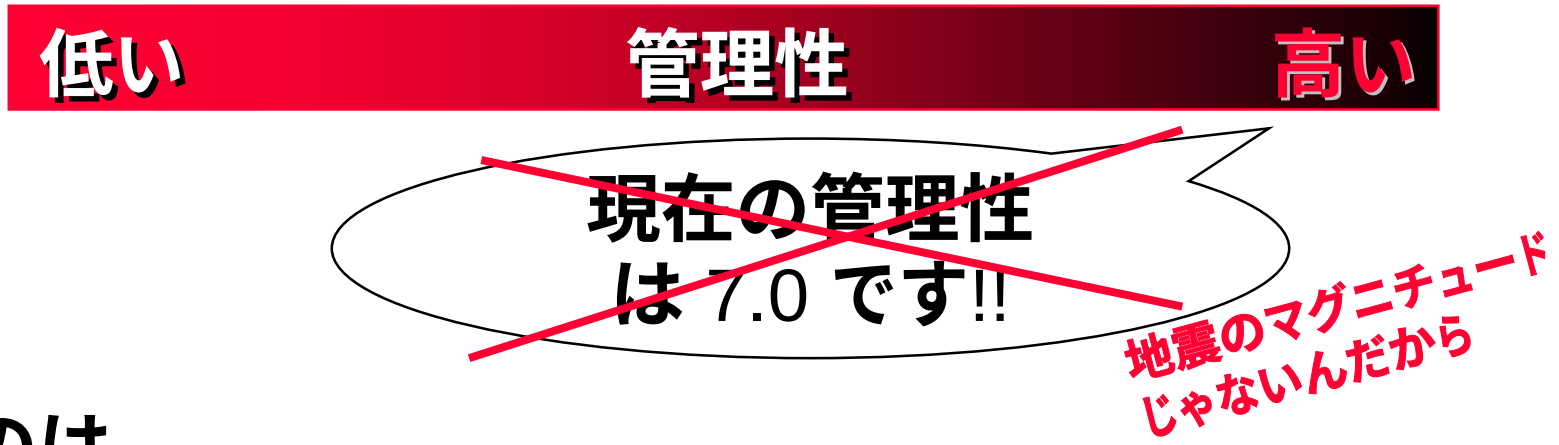
アプリケーションの管理性とは?

- 数値が高いほど管理しやすい
- 数値が低いほど管理しづらい

と一般的に言われており、
システム運用の責任者が気にする指標

アプリケーションの管理性とは?

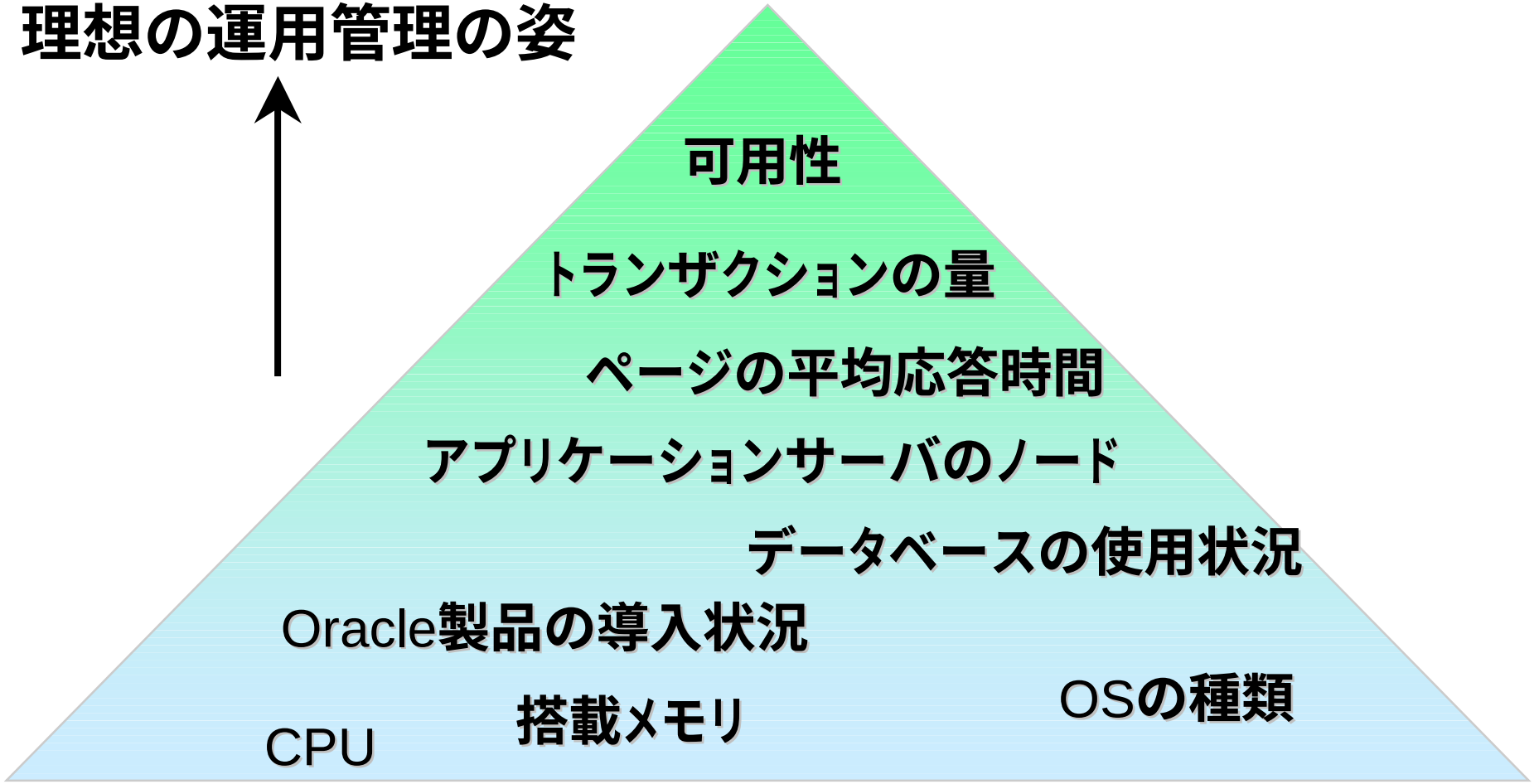
- でも、数値化された指標は一切存在しない



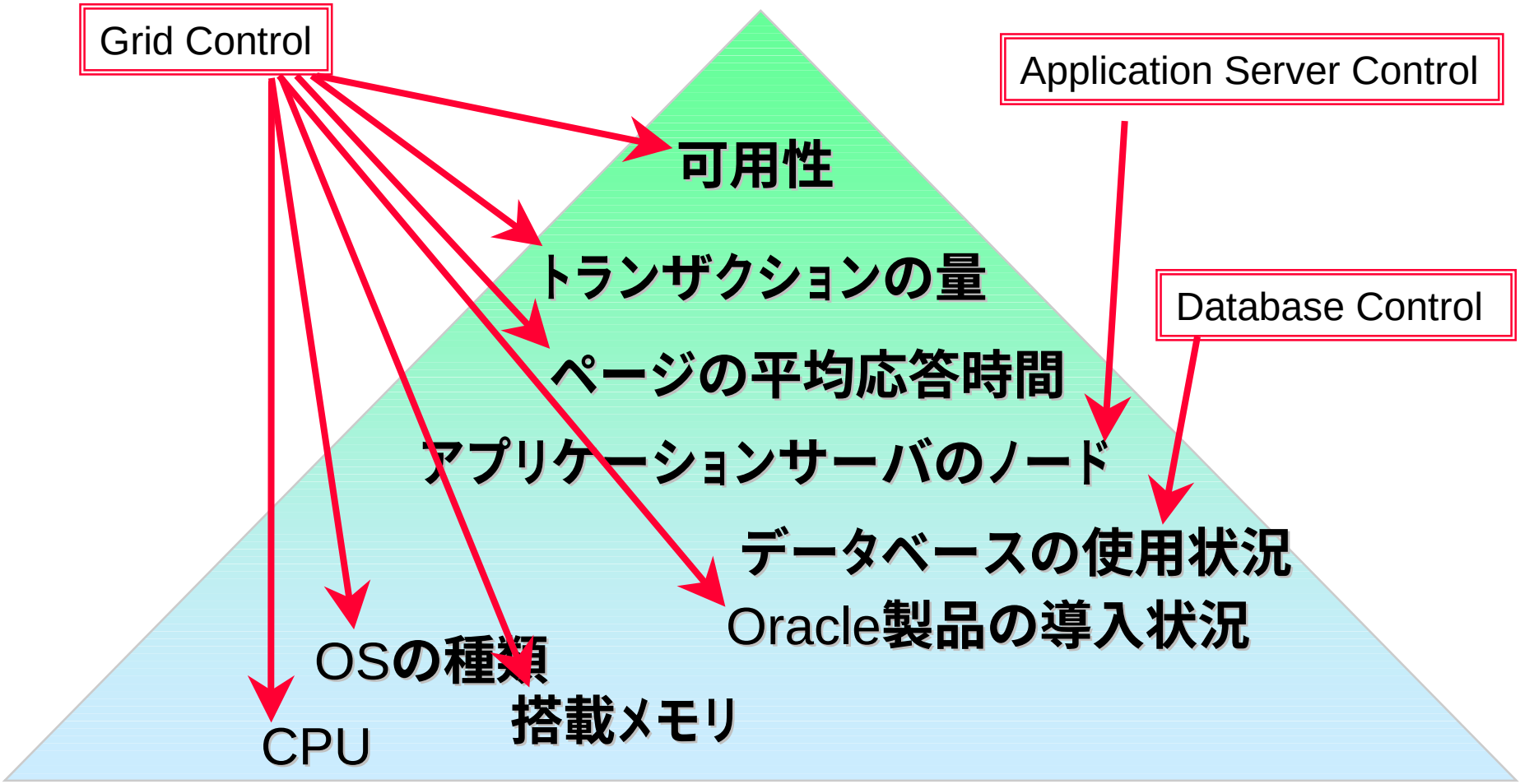
あるのは、
管理ツールが使いやすいか、わかりやすいかという
感覚のみ

管理したいものは何?

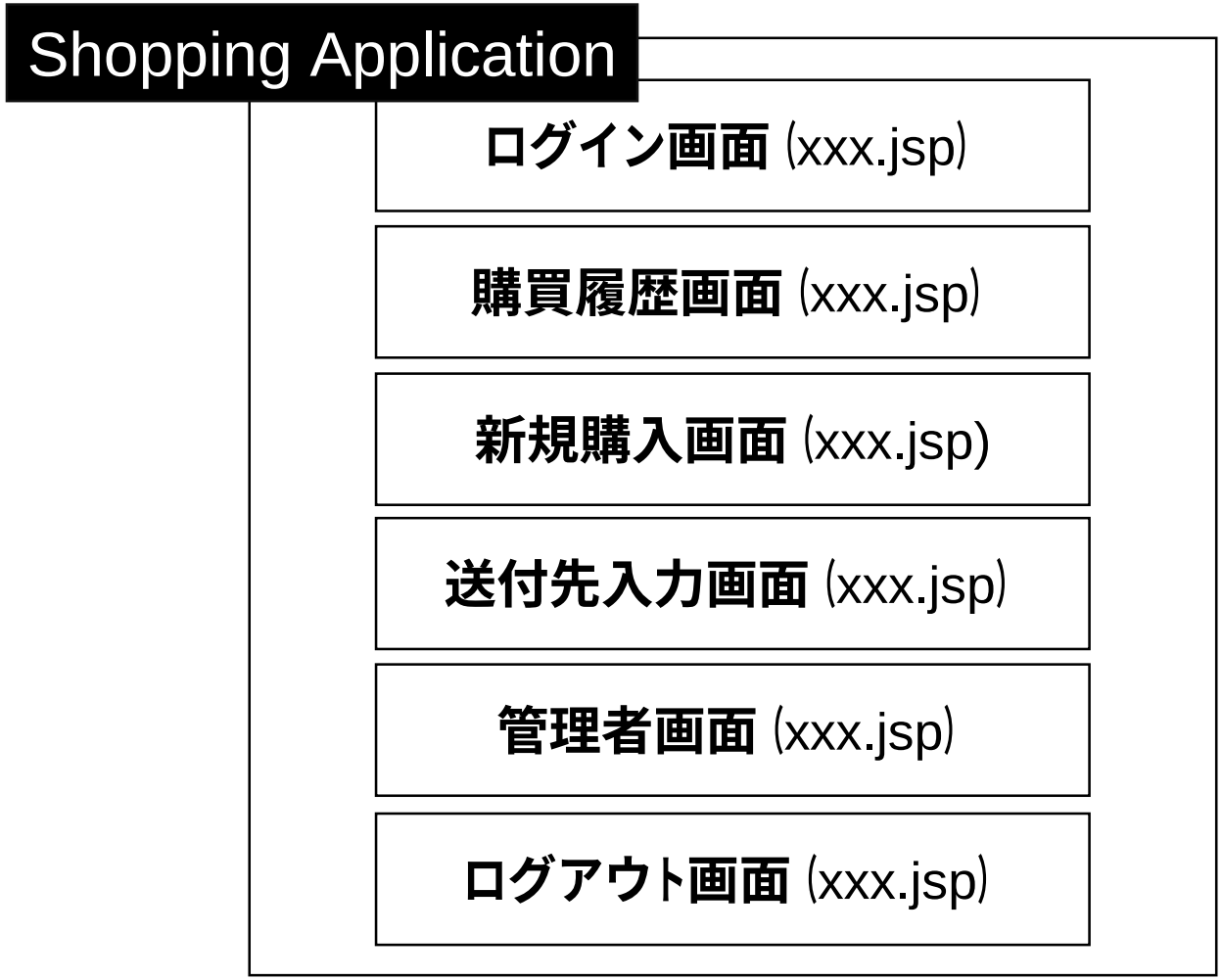
理想の運用管理の姿



管理したいものは何?


Grid Control
Application Server Control
Database Control
可用性
トランザクションの量
ページの平均応答時間
アプリケーションサーバのノード
データベースの使用状況
Oracle製品の導入状況
OSの種類
搭載メモリ
CPU

Webアプリケーションの概念の例



Webアプリケーションと
実際の監視対象のギャップ

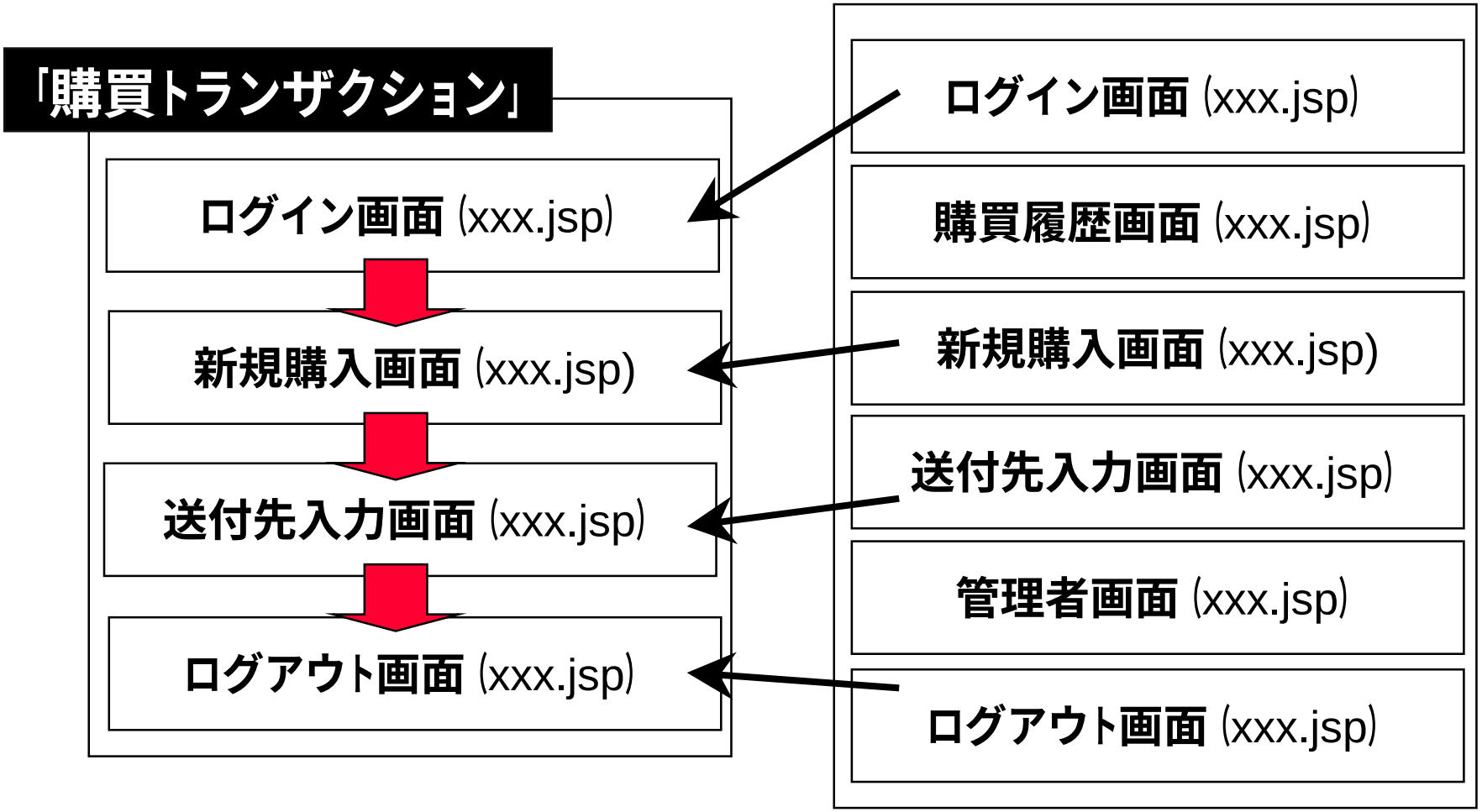
管理者はそれぞれのURLがどのくらい呼ばれたとか、CPUをどのくらい利用したとか、メモリをどのくらい占有したか？
を知りたい以上に、実際のユーザ操作がストレス無く行われているかを知りたい!!

であれば、実際の操作を監視する
＝ ビジネストランザクションの監視

ビジネストランザクション

- ログイン、商品購入、支払、納品先指示などの一連のWebブラウザによるアクション
- Grid Control は一定の間隔で実際にこのビジネストランザクションを行い、その結果を記録
  - アクセスエラーやレスポンス低下はアラートとしてすぐに報告

ビジネストランザクションの概念の例



ビジネストランザクション

- ビジネストランザクションはWebブラウザから行うすべての一連の処理を記録
  - J2EEアプリケーション
  - 静的Webページ
  - リンククリック
  - 値の入力
  - …

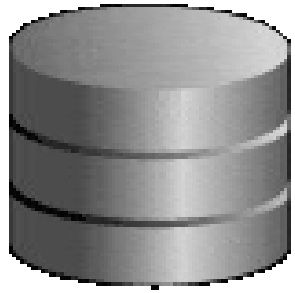
とっても意外な、ビジネス トランザクションの管理方法

このボタンを押すと空のWebブラウザが起動し、すべての操作を実際に行う

＝ 操作は自動的に記録される

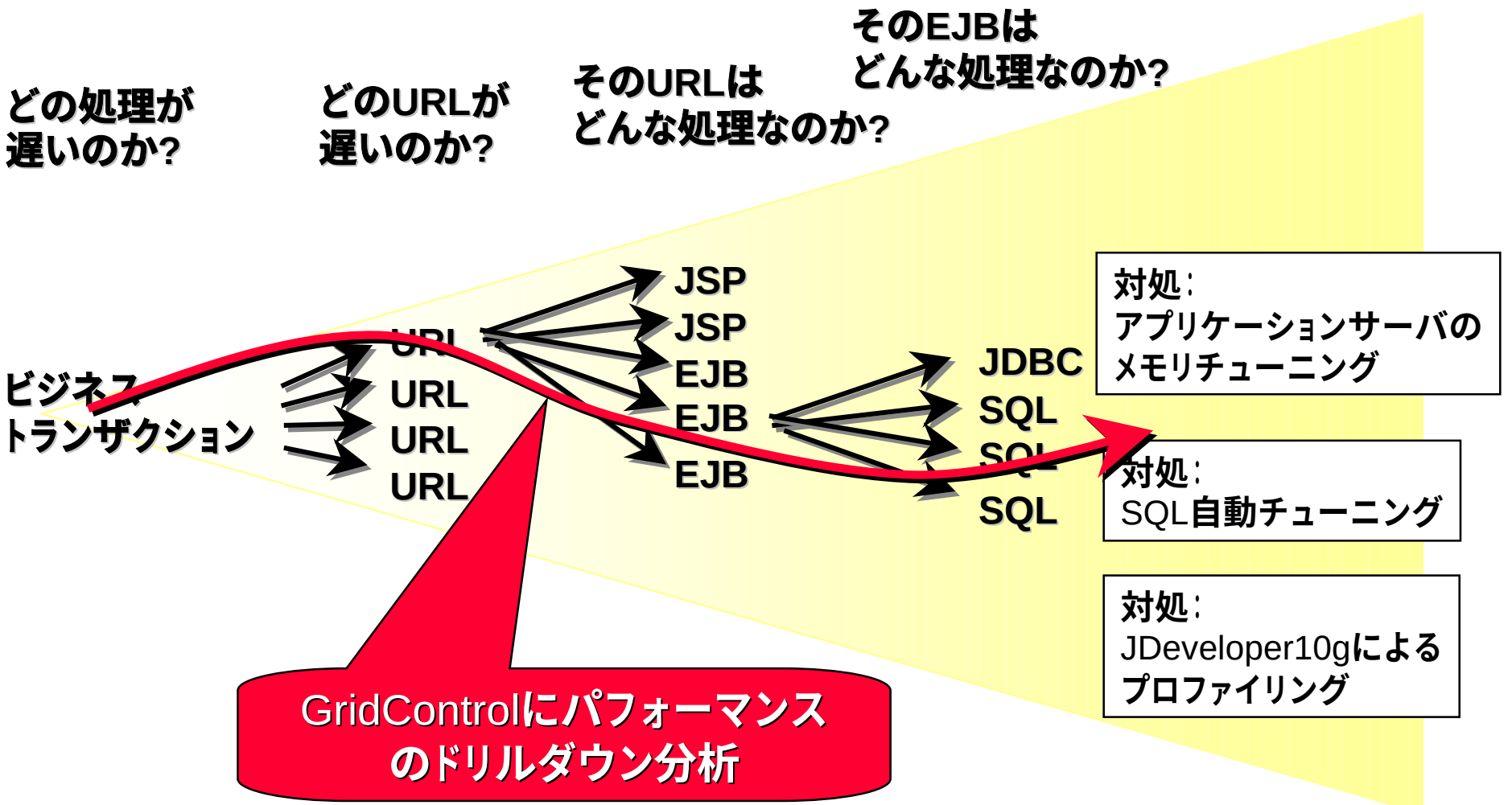
パフォーマンスチューニング方法

- パフォーマンスボトルネックはどこに潜んでいるかわからない…



テクノロジ	JSP/Servlet EJB…	PL/SQL
	J2EEアプリケーションが悪い? APサーバのチューニング不足?	SQLが悪い？ DB設計が悪い? PL/SQLが悪い?

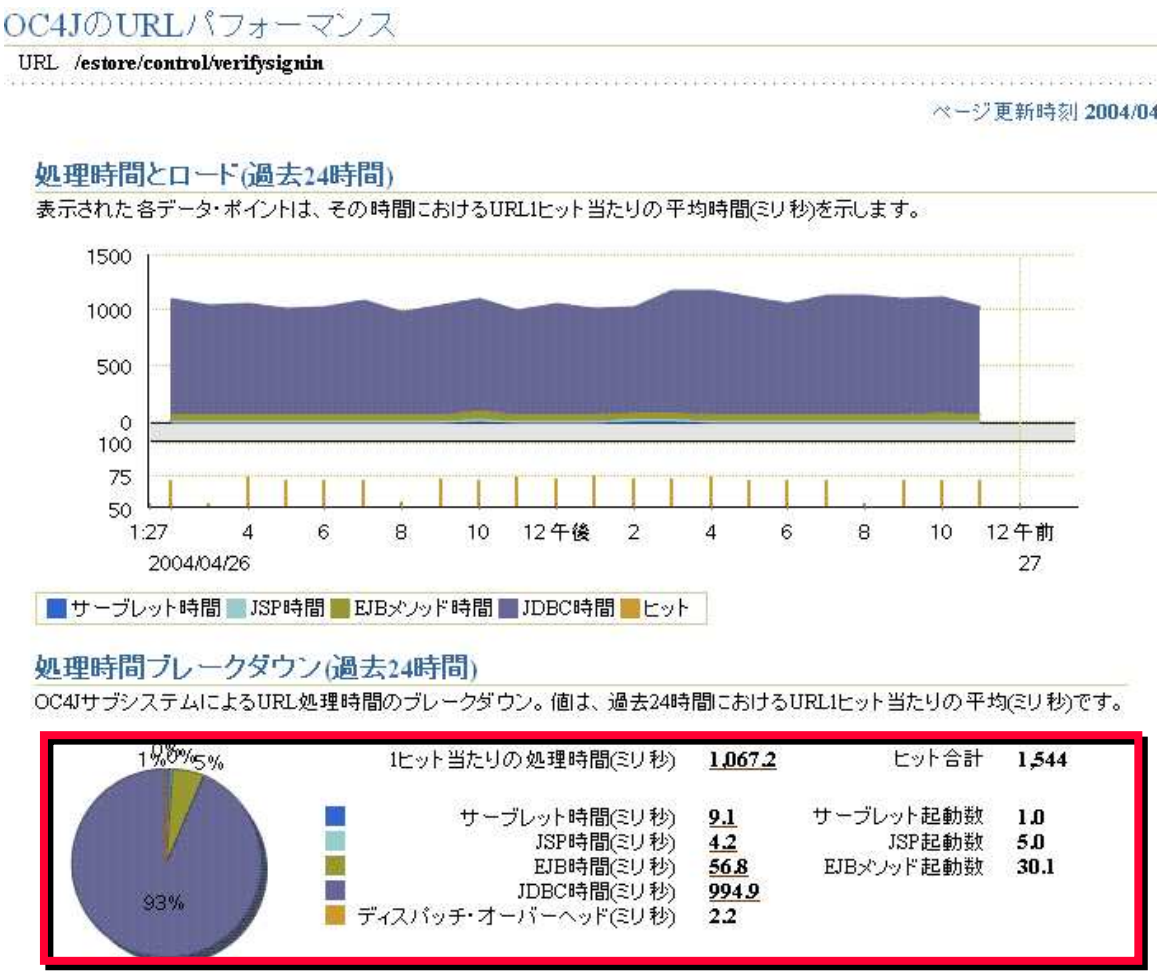
パフォーマンスチューニング方法



Step1:
ビジネストランザクションのアラート

Step2:
どのURLがボトルネックか?

Step3:
そのURLはどんなテクノロジーか?



Step3:
どんなSQL文がボトルネックか?

JDBC接続

1ヒット当たりの接続数 **14.0**　1ヒット当たりの接続時間(ミリ秒) **905.2**

SQL文を表示

すべて選択 | 選択解除

選択	データ・ソース	スキーマ	リクエスト率	1ヒット当たりの接続数	1ヒット当たりの接続時間(ミリ秒)	SQL時間(ミリ秒)
☑	使用不可	使用不可	84.8	14.0	905.2	89.8

選択した接続に対して実行されたSQL文

1ヒット当たりのSQL実行(ミリ秒) **56.0**　1ヒット当たりのフェッチ **24.0**
1ヒット当たりのSQL実行時間(ミリ秒) **89.6**　1ヒット当たりのフェッチ時間(ミリ秒) **0.1**

SQL文	データ・ソース	スキーマ	リクエスト率	平均起動数	1ヒット当たりのSQL実行時間(ミリ秒)
SELECT VALUE FROM NLS_INSTANCE_PARAMETERS WHERE PARAMETER =' ...	使用不可	使用不可	3.8	42.0	40.8
UPDATE profile SET langpref = ?,favcategory = ?,mylistopt = ...	使用不可	使用不可	0.8	2.0	8.6
SELECT userid,langpref,mylistopt,banneropt,favcategory FROM ...	使用不可	使用不可	0.7	3.0	7.8
SELECT userid,status,email,firstname,lastname,addr1,addr2,ci...	使用不可	使用不可	0.6	2.0	6.8
UPDATE signon SET password = ? WHERE username = ?	使用不可	使用不可	0.6	1.0	5.9
SELECT username, password FROM signon WHERE username = ?	使用不可	使用不可	0.5	2.0	5.1

ORACLE JDEVELOPER 10g

ORACLE®