Ruby でシステム開発を加速しよう！！

魅惑の珍獣

はっし～～ ™

Agenda

- なぜRubyを開発ツールとして使うのか
  - Java連携
    - Rjb，JRuby
  - DSL
    - 言語内DSL，学習コストの低さ
  - ツール開発の生産性
- 開発フェーズでのRubyの使いどころ
  - 要求開発／要件定義フェーズ
  - 設計フェーズ
  - ☆ 開発フェーズ ☆
  - 運用フェーズ
- まとめ

Javaとの連携

- JavaEE用のツールはJavaとの連携機能が必要
- というか、連携できないと使えない・・・・

- Rjb（Ruby Java Bridge）： 最新版は1.0.3
  - arton先生作
  - Rubyに対するJavaブリッジ
  - 環境ごとにスレッドモデルが異なるのでGUIには向かず

- JRuby（最新版は 0.9.8）
  - Java製Ruby1.8.5互換のスクリプト言語
  - Charles Nutter氏，Thomas Enebo氏（Sun）
  - Javaのスレッドモデルと同一
  - BSF／JDK6 Scripting Support

Jruby現在の進捗(2007/1/31段階)

- Railsサポート（テスト通過率）
  - MySQLとの組み合わせは100% 通過
  - 他のDBとの組み合わせはこれから。
- パフォーマンスの改善
  - Ruby→Javaメソッドのキャッシュ（2割ほど速度UP）
  - jrubyc：RubyスクリプトからJavaバイトコード生成
    - 期待する成果の3割程度の出来
    - メソッドレベルでclassファイルの生成
    - ただし、クロージャー，クラスなどの扱いはこれから
  - JRubyによるYARVバイトコード・サポート
    - YARV（Yet Another Ruby VM）・・・Ruby2.0の標準VM

DSL（Domain Specific Language）

- DSLって何？
  - 『特定用途』向けの小さな記述言語
    - 例）ビルドツール，設定ファイル，Rails（Webアプリ用DSL）
    - SQL，JSP，EL(式言語)，XML もある意味DSL
- Rubyを使った言語内DSL
  - いちいちDSL用のコンパイラを書いたら大変！
    - DSLごとに文法を覚えるのも面倒
    - ⇒ だったら、Rubyの文法のサブセットを使おう（言語内DSL）
    - ⇒ RubyをDSLに使用するのでツールに一貫性が出る！

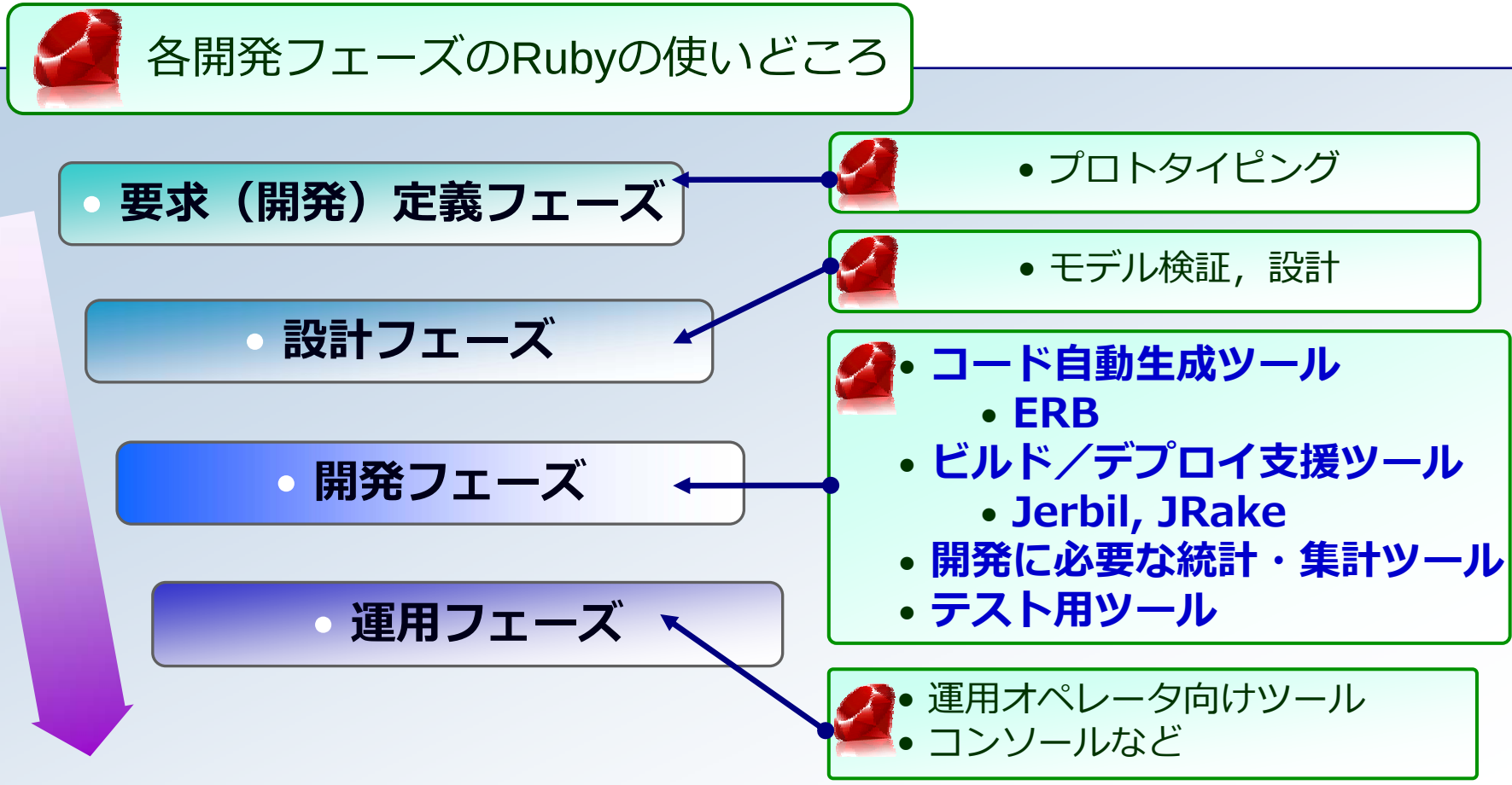
DSL（続き）

- DSLの特徴
  - 宣言的（Javaのimport文，アノテーションは宣言的）
  - 構造化による名前空間の保護（XMLを考えよ！）
  - 単純な条件分岐 / 繰り返し文法を備える

- DSLをRuby言語内DSLにすると・・・
  - コンパイラを書く(XMLをパースする)手間が省ける
  - 強力なリテラル(配列/ハッシュ/文字列/正規表現)
  - 強力なライブラリが使える。
  - Rjb／Jrubyを使ってJavaライブラリも呼べる
  - なにより文法の一貫性が保てる。
    - Rubyの文法を知っていれば想定内で記述できる

開発フェーズでのRubyの使いどころ

- Java連携，DSLという道具を得て、 要求開発～ 運用までの使用を考えてみましょう。



要求開発フェーズ

- フロントローディングでの使用
  - 本番開発が別言語だと・・・
    - アーキテクチャの早期確定の側面が犠牲になる
    - システム構造評価プロトタイプには使えない？
  - Ruby on Railsの使用を考える（以下で効果的？）
    - プレゼンテーション支援プロトタイプ
      - 複雑なビジネス要件の実現イメージを早期に確定
    - ToBe業務評価プロトタイプ
      - 操作イメージを含めた新システムの実現イメージを早期に確定
    - ※「このまま納品して！リスク」が出ますが・・・
  - Railsによってデータモデルの成長が可能
    - Database Refactoring
    - 本番へのデータモデルの流用は可能。

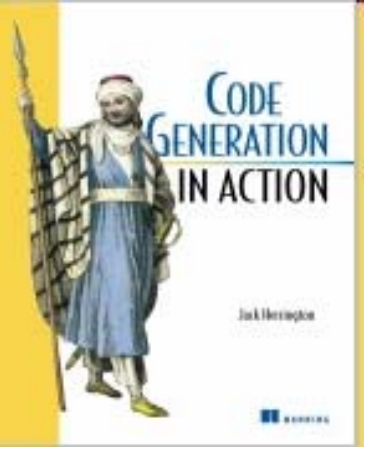
設計フェーズ

- 設計フェーズでの使用
  - 約束の地（RubyAsSketch）
    - UML初心者時RubyからよくUMLを起こした。
    - シーケンス図を書くならRuby（Javaは面倒だった）
    - デザインパターンの動作検証
    - Action Semantics や OCLを書くならRubyの方が楽
    - 本当に正しいモデル・チェッキングをしたいなら、形式言語を使いましょう。

開発フェーズ

- Rubyを利用して開発生産性を向上させる

- **例１）コード自動生成ツール**
  - 自動生成ツールは必要（特にDao層）
    - 既存ツールはオレ流に適合しない場合が多い
  - Code Generation in Action



- 例２）ビルドツール
  - Jerbil・・・Rjbを利用
  - Jrake／Raven → 統合へ
- 例3）テスト用ツール
  - 例）
    - RDB用のダミーデータ作成
    - ソケット通信用ダミーサーバ

チューニングフェーズ

- RDB関連ツール
  - RDBの統計情報を定期的に取得するツール
    - グラフ／チャートツールを利用する（Ziyaなど）
    - Webアプリを使ってSQLを流すツールなど
    - 定期的（日次、週次）なレポートを生成
  - 高負荷アクセス進捗状況表示ツール
    - 巨大テーブルのフルスキャンや大量のソート処理など、SQL実行に

    時間がかかる場合の終了時刻の予想をするツール

運用フェーズ

- 運用オペレータ向けツール
  - シェルなどバラバラの言語のものをRubyで統一
  - Java APPサーバのコンソール
    - 再起動，アラートなどオペレータが使いやすいI/F。
  - 運用向けログ・フィルタリングツール
    - オペレータがどうするかを分かりやすく表示

納品物としてのRubyを考える

- JRuby on Rails
- JavaEE App Server上でのJRuby on Railsの動作
  - ActiveRecord JDBC / rails-module
  - AsyncWeb / Grizzly
  - JRubyによりEJBコンポーネントの呼び出しが可能

- 企業内Webアプリは基幹系との接続が重要。
  - 基幹系接続部分のみにEJBコンポーネントを使用する
    - EJB3 + JMS1.1 で 2 Phase コミット も視野に入る
  - ESBへの非同期／同期接続という選択肢もあり

納品物としてのRuby … cont.

- 今後、スクリプト言語のJavaEEでの使用が増加
- 鈴木雄介氏の３つの分類
  - アウターゼリー
    - 開発ツールとしての用途（これまで述べてきた内容）
  - インナーゼリー
    - アプリケーションの内部での使用（ビュー層）
  - スーパーグルー
    - コンポーネント同士の結びつけ（BPEL）
- Webアプリ自体がスーパーグルーと見る
  - これまでのホストとの接続が好例
  - 今後はEJBやESBへの接続でミッションクリティカル領域を担保。
  - Rubyは有用な選択肢と見る。

まとめ

- Java連携（Rjb，JRuby）と DSL がツールのベース
- 開発プロセスへの適用方法
  - 要求開発： フロントローディングでのRubyの使用
  - 設計フェーズ：Rubyを道具とした設計
  - 開発フェーズ：ビルドツール，コード生成に使い途
  - 運用フェーズ：オペレータ向けツール
- 納品物としてのRuby
  - JRuby＋JavaEE App Server の利用がこれから始まる