PROJECT-R
Ruby X TECHNO PROJECT

～Rubyが切り開くイノベーション ～
（新システム分野と開発標準）

2011年7月22日

島根県情報産業協会　会長　吉岡宏

Ruby業務システム開発の実績

2007年　IPA(情報処理推進機構)実証事業

IPAオープンソフトウェアセンターがRuby/Railsを基幹業務に適用する実証事業を実施。

⇒　松江市殿)医療・介護高額合算システム

(日本初:バッチ処理を含む基幹業務全体)

約50人月　　2007．9～2008．3

オン:14本、バッチ:24本

業務開発チーム:6名

㈱テクノプロジェクト、㈱マツケイ

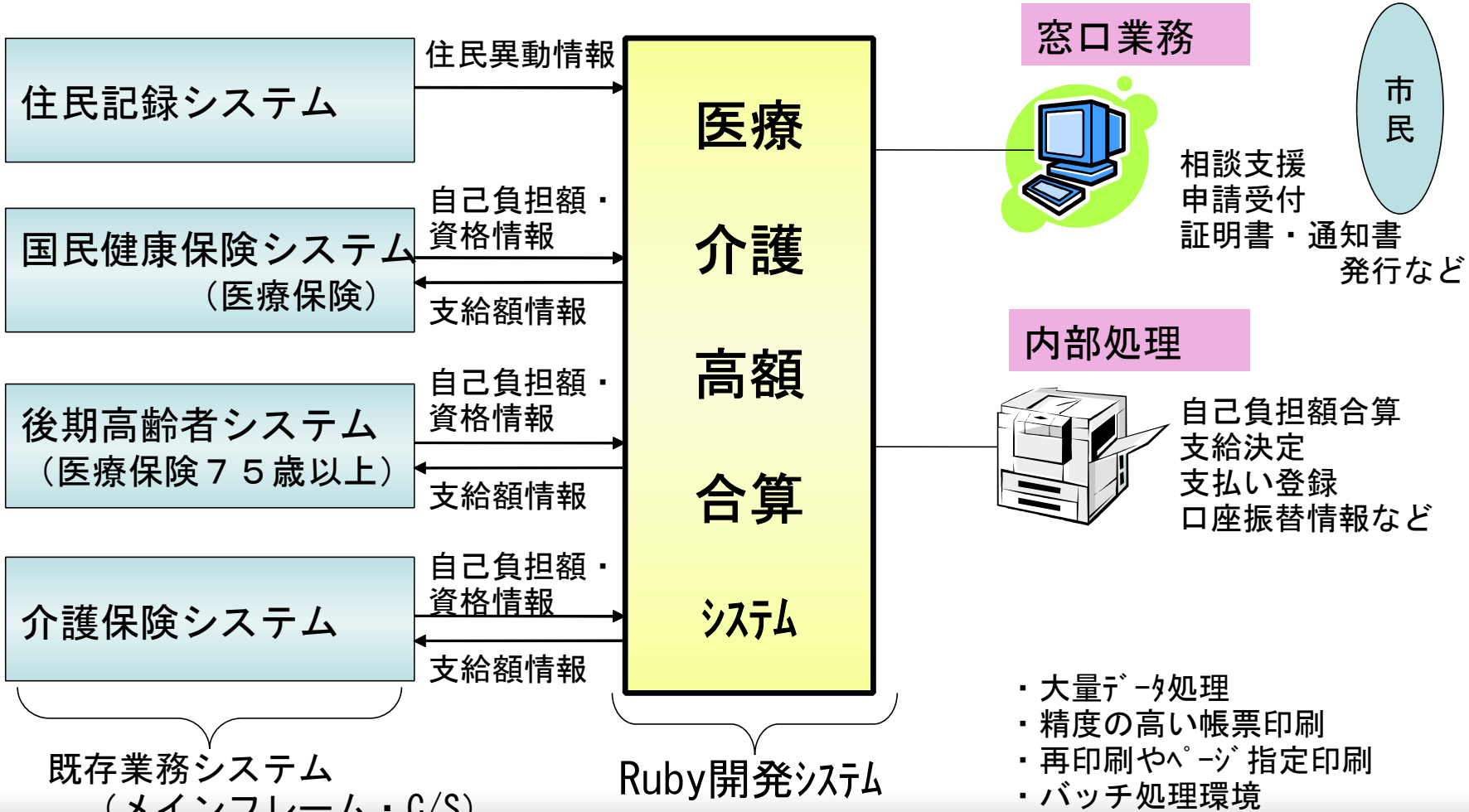
共通技術チーム:3名

㈱ネットワーク応用通信研究所

医療・介護 高額合算システム

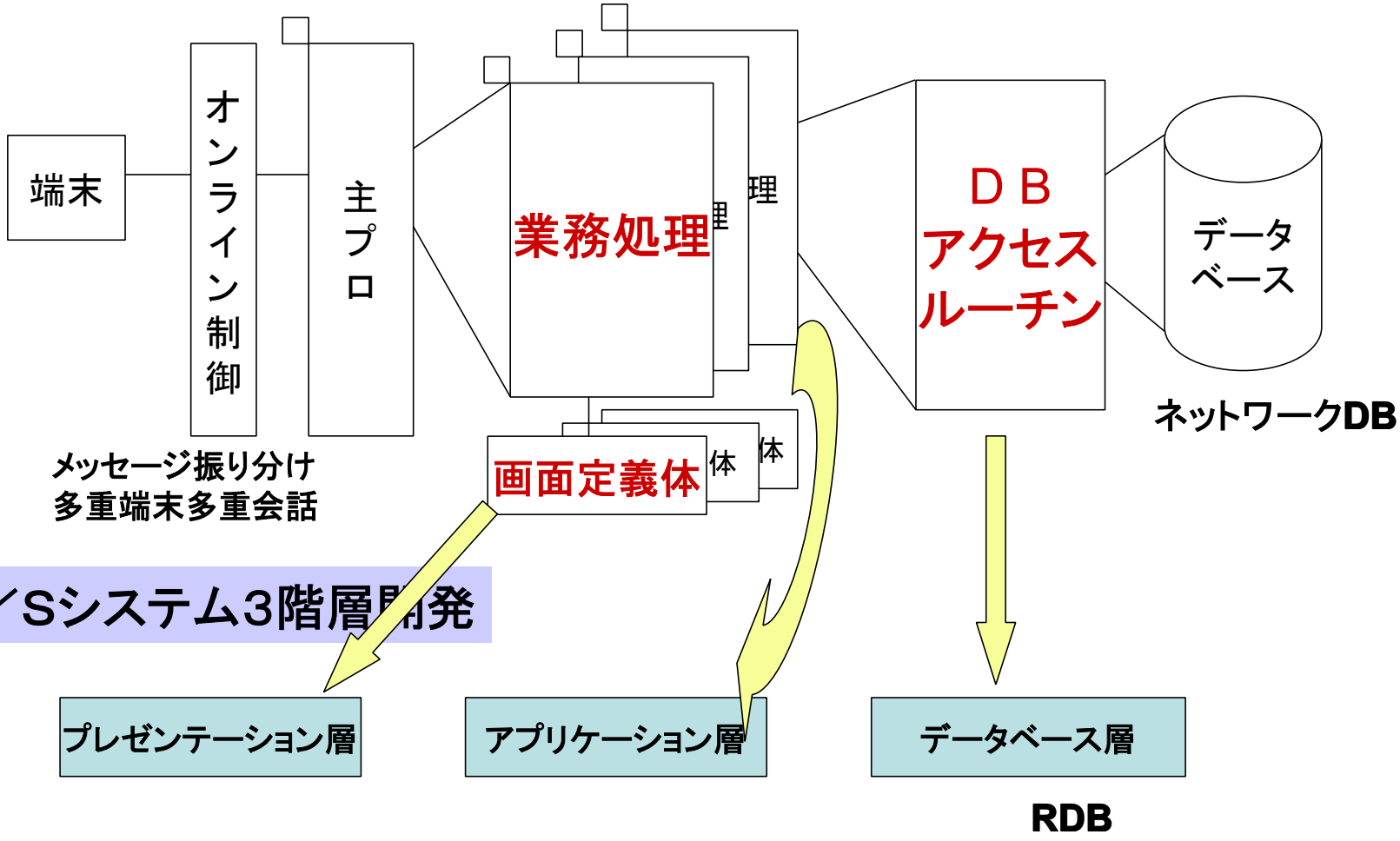
PROJECT-R
Ruby X TECHNO PROJECT

2007年度 IPA公募事業　医療・介護の高額合算システム（松江市殿）



メインフレームのオンライン構造 （MainFrame）

PROJECT-R
Ruby X TECHNO PROJECT

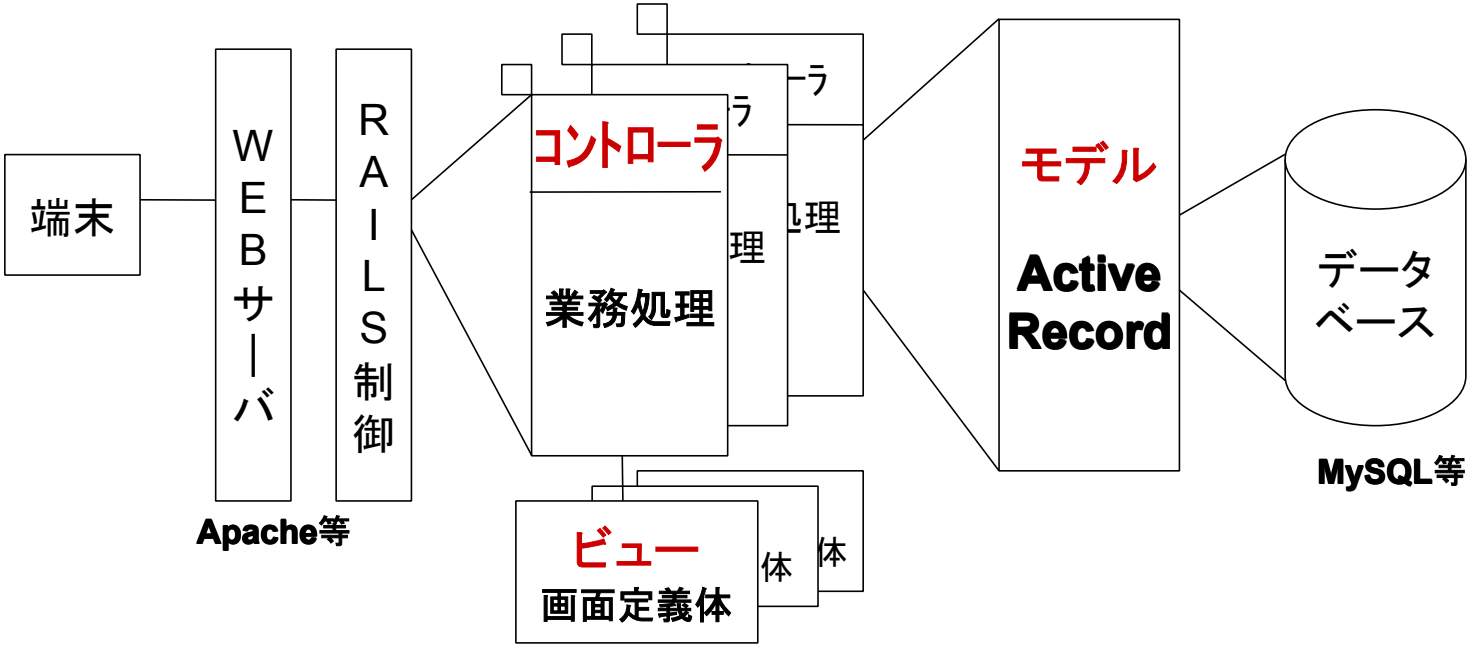


業務処理に集中して開発できる環境　**Best System Structure**

Webアプリ（Railsシステム構造）

PROJECT-R
Ruby × TECHNO PROJECT

Ruby　on Rails（機能分割：ﾓﾃﾞﾙ・ﾋﾞｭｰ・ｺﾝﾄﾛｰﾗ）

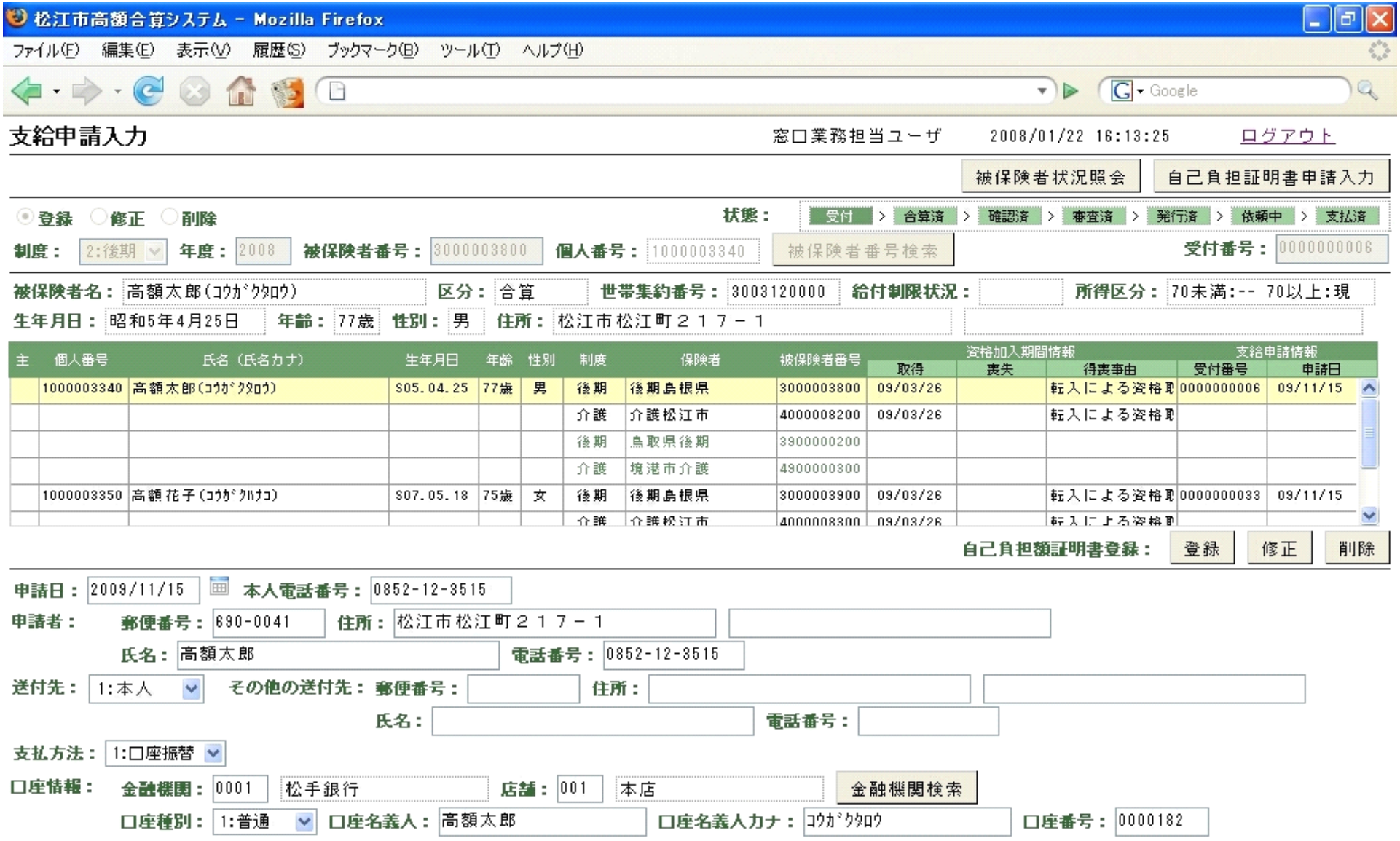


※Ruby on Rails

M：モデル（DBマクロ）　検索・挿入・更新・削除のSQL　（一部業務も含む）

V：ビュー（画面定義）　HTML・ﾍﾙﾊﾟｰ ・・・HTMLｴﾃﾞｨﾀｰ（Webﾃﾞｻﾞｲﾅ）

C:コントローラ（オンライン制御・業務処理）・・・業務ロジック



Railsでも、C/Sシステムと同様な画面IF（コンボボックス、オプションボタン、サブウィンドウ等）

窓口に市民が来た時の対応　⇒　1つの画面で完結（画面遷移の少ない処理）

（上段：申請者情報,、中段：受給履歴、下段：申請内容）

生産性（２００８．２）

PROJECT-R
Ruby X TECHNO PROJECT

前提条件	全工程(100%)	設計工程（基本設計、詳細設計）	30%
		製造工程（PG設計、プログラミング、単体テスト）	40%
		テスト工程（結合テスト、総合テスト、運用テスト、操作研修）	30%

Ruby・Rails と既存の基幹業務の開発方法との生産性比較

比較方法：　初めてのRails開発経験者に、ヒアリングを実施。
　　前提として、Rails開発を2～3回こなし、習熟した場合に、生産性はどのようになるか。
対象者：　50代SE　2名、30代SE/PG　3名
　　（COBOL,VB,Javaによる基幹業務の開発経験者）

設計工程(30%)＋製造工程(40%)×3/4＋テスト工程(30%)　⇒　全工程(90%)・・・10%改善
変わらない　　25%改善　　変わらない

- ウォータフォール開発、開発要員は当初から全員投入しRuby教育やプロト開発を実施する。
- ドキュメント　設計工程：基本設計/詳細設計書、テスト工程：テスト仕様書/成績書
- 詳細設計は、プログラム単位に詳細な機能説明を記述。プログラムのロジックを記述する
  プログラム設計書は作成していない（比較の元となる開発もプログラム設計書：未作成）。
- 基幹業務の特性として、チェック処理が多い。1つのデータ登録に、100ステップ程度の
  整合性チェックなどが頻繁にある。→ どの言語で開発しても、生産性の改善はできない。
- 自動テスト機能は、使用していない。アジャイルのような繰り返し開発に有効であるがウォー
  タフォール型の開発では、テスト手順の登録作業に、かえって工数が増大すると考えた。

開発標準：COBOLerの復活（2007～）

ベテランSE（COBOLer）がWebシステムの開発に参加できる

①ウォータフォール開発　（設計⇒製造⇒テスト）
②厳格なコーディング規約　（Rubyの自由度をなくす）
③日本語のDB項目名称（設計書とソースの見易さ）
④SQL記述（判り易さ、但し便利な部品が使えない）
⑤バッチ処理も同一言語で開発
⑥構造化設計（業務ロジック重視、従来のﾄﾞｷｭﾒﾝﾄ類）
⑦少ない部品（少量の部品から始める、覚え易さ）
⑧画面定義（HTML）は、専任化：Webﾃﾞｻﾞｲﾅｰ
⑨クラス設計：ﾃｰﾌﾞﾙ毎の業務ﾛｼﾞｯｸを含めたｱｸｾｽﾙｰﾁﾝ

COBOLer向けRuby／Rails教育コース　by　テクノプロジェクト

開発システムの事例（2007～2010）

県内自治体	遺失物管理システム	約40人月
県内自治体	行政システム	約35人月
県内自治体	施設予約システム	約20人月
県内財団法人	施設管理システム	約12人月
医療機関	勤務管理システム	約18人月
医療機関	化学物質管理システム	－
金融機関	ホームページ	約4人月
県外ソフトウェア業	業務系システム（パッケージ開発）	約20人月
県外ソフトウェア業	Railsを用いたシステム開発支援	－
県外情報サービス業	ソフトウエア構成管理システム	約50人月
経済産業省	地域経済情報化基盤整備補助事業 （地域連携：島根2社＋広島2社）	－
しまね産業振興財団	教育講座 業務開発者のためのRuby/Rails実践	－
島根県	母子寡婦福祉資金システム（iOFW）再開発業務 心身障害者扶養共済システム（iOFW）再開発業務	約40人月

アジャイル開発（2010）

PROJECT-R
Ruby X TECHNO PROJECT

Rubyビジネスモデル研究実証事業

（製造業向けアジャイル開発）

- 生産ラインの各種機械の検査記録システム

  顧客情報システム部門：1名、SE/PG：3名

  開発期間（6カ月）と予算（コスト）確定

  システム仕様は作りながら決定

  目標レベル設定、但し顧客満足度優先 ⇒ 結果：未達 → その後顧客側で完成

- アジャイル（スクラム方式）

  定期的な顧客リリース（2カ月単位） ⇒ 仕様調整

  開発チーム内は2・3週ごとに結合テスト ⇒ 早い段階で整合性確認

  テスト駆動開発（テストファースト+リファアクタリング）

  ペアプログラミング ⇒ 見られるという緊張感

  ダイナミックな作業割り振り ⇒ 前提：開発者の業務理解度が平準

  朝会・夕会 ⇒ 基本10分、時々議論の場にもなる

ウォータフォール ➡ アジャイル

PROJECT-R
Ruby X TECHNO PROJECT

生産性と品質

※ 良いものがオープンソースで提供される
Ruby＋Rails環境を前提にした米国製品

・道具の活用（ﾃｽﾄﾂｰﾙ・課題管理・ｿｰｽ管理・ﾃﾞﾌﾟﾛｲ）
・部品の活用（GEMﾊﾟｯｹｰｼﾞ・Railsﾌﾟﾗｸﾞｲﾝ・社内部品）

⬇

プログラミング作業　加工　⇒　組み立て
テスト駆動開発（テストの自動化＋リファクタリング）

生産性向上　＋　品質向上

開発標準を目指して

PROJECT-R
Ruby X TECHNO PROJECT

アジャイルを業務システム＆請負い契約に取り込む

開発手法

①開発方式とドキュメント
②開発チーム
③テスト駆動開発
④ペアプログラミングとソースレビュー

ツール

⑤プロジェクト管理・開発ツール
⑥外部ライブラリィの活用
※帳票作成ツール

品質・生産性・顧客満足度・開発者満足度　全ての向上に向けて

開発手法

PROJECT-R
Ruby X TECHNO PROJECT

①開発方式とドキュメント

（1）設計：ウォータフォール

業務システムは、一般的に制度・法律・慣習が詰まった複雑なもの。

DBアクセスでは、多くの更新系処理が発生する。イレギュラー処理。

⇒　設計が重要、特にDB設計

最終ﾄﾞｷｭﾒﾝﾄ：ﾒﾝﾃﾅﾝｽ手引書として整備（含む納品物）

構成品管理表、基本設計書・詳細設計書、操作・運用ﾏﾆｭｱﾙ

※開発途中は、ホワイトボードのデジカメ化でもOK

（2）実装（PG設計～結合テスト）：アジャイル＝素早く、繰り返し

⇒　繰り返しの中で、早め早めの顧客確認（満足度）と開発チームの整合性確認

※顧客リリースは2～3カ月の単位で段階的に行う。

実装は2～3週単位に。進捗と品質の確認を行う。

開発手法

PROJECT-R
Ruby X TECHNO PROJECT

②開発チーム

短期開発（1年以内）：最初から全員投入

⇒ チームワーク・コミュニケーション

PGの役割：SE補助作業・業務習得・先行開発

“若手エンジニアの業務ノウハウ修得の場”

朝会（ｽﾀﾝﾄﾞｱｯﾌﾟﾐｰﾃｨﾝｸﾞ）：短時間→一人一人の進捗と予定の報告

× ダイナミックな作業割り振り（朝会で調整。前提：全員が同一な業務ノウハウ）

複雑な業務では、不可能である。

③テスト駆動開発

テスト手続きの登録＆自動テスト（含む、ﾃｽﾄﾃﾞｰﾀ登録）

効果：品質向上（テストの自動化＋ソースの最適化→リファクタリング）

単体・機能テスト（ロジックの内部仕様）　評価：△

活用：ホワイトボックステスト（ﾓﾃﾞﾙ・ﾋﾞｭｰ・ｺﾝﾄﾛｰﾗ・ﾓｼﾞｭｰﾙ単位）

繰り返し[1ﾃｽﾄｹｰｽ登録 → ﾛｼﾞｯｸ記述（数ｽﾃｯﾌﾟ）→ リファクタリング]

機能テスト（業務の外部仕様）　評価：◎

活用：ブラックボックステスト → ユーザサイドのテスト

レグレッションテストの容易性、ライフサイクルでの活用　⇒　品質向上

開発手法

PROJECT-R
Ruby X TECHNO PROJECT

④ソースレビューとペアプログラミング

（1）基本パターン

ソースレビュー　コーディング規約遵守、第三者の目に触れることが重要
タイミングは単体テストの終了段階

開発者：中級レベル以上は単独開発が基本

（2）オプション

ペアプログラミング

開発者：初級レベル（**Rails**初心者、新入社員アソシエイト、業務初めて）

→ 早期立ち上げを目的に、中級レベルのエンジニアがトレーナ役を担当し、つまづきを最小化する
初心者が質問しやすい環境を強制的に作る

ツール

PROJECT-R
Ruby X TECHNO PROJECT

⑤プロジェクト管理・開発ツール

Ruby：1.9 + Rails:3.0 （ Ruby:1.8 + Rails:2.3)
Arel(DB),CSS(画面)

Redmine 案件管理・ToDo

Subversion ソース履歴管理（Redmineと連携）

NetBeans 開発ツール（IDE＋ｿｰｽｴﾃﾞｨﾀｰ）

Cucumber 結合テストツール（画面単位、外部：ﾕｰｻﾞｻｲﾄﾞﾃｽﾄ）
日本語によるテスト仕様記述

Capistrano デプロイ（本番環境への更新モジュール反映）

※データベース

新規開発 → MySQL・PostgreSQL
既存ｸﾗｱﾝﾄｻｰﾊﾞｼｽﾃﾑのWeb化 → Oracle・SQLSeverの継続使用
複合キーのサポート部品あり

ツール

PROJECT-R
Ruby X TECHNO PROJECT

⑥外部ライブラリィの活用

便利な部品の活用（ネット上＋社内実績）

部品の活用度に応じて、生産性・品質向上

※ 部品の管理が重要

活用ﾉｳﾊｳ・逆引き・更新情報・ﾗｲｾﾝｽMIT・実績プロジェクト

利用者・使用上の注意事項・Ruby/Railsのﾊﾞｰｼﾞｮﾝ　等

例：Pagenation 帳票・画面の一覧表および次ページ送りの部品

ライブラリィ ： GEMパッケージ、Railsプラグイン

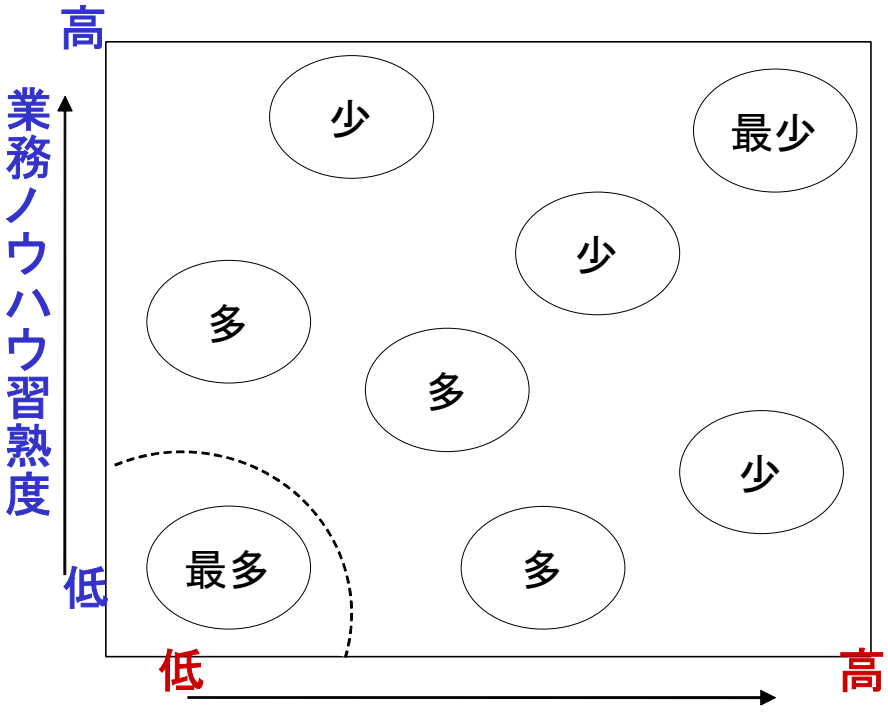
※帳票ツール

- Java用：JasperReports（iReport）・・・**Ruby**より利用可能
- Ruby用：ThinReports（帳票ｴﾃﾞｨﾀ、ﾋﾞｭｰｱ：PxDoc）
  - ｵｰﾌﾟﾝｿｰｽで㈱ﾏﾂｹｲより提供（**TPJ**グループ企業）
  - 将来構想：HTML5ﾋﾞｭｰｱ開発・PDF化機能

繰り返し回数の最適化

PROJECT-R
Ruby X TECHNO PROJECT

繰り返し頻度のグラフ



プロジェクトチームの開発経験度
（Ruby/Railsｱﾌﾟﾘ、新技術を含む）

最少：プロト開発
設計工程での実画面相当の画面を
作成し、顧客確認（操作性）
１回（設計＋実装）

少　：スパイラル開発、サブシステム単位
設計１回 → 実装２～３回
基本設計１回
→ ２～３回(詳細設計＋実装)
※１回目は主要画面中心

多　:アジャイル開発
基本設計1回
→ n回(詳細設計＋m回(実装))

最多:アジャイル開発　＝　無謀

※ 実装：ﾌﾟﾛｸﾞﾗﾐﾝｸﾞ・ﾃｽﾄ・結合ﾃｽﾄ＋顧客確認

第三の軸

顧客要件や利用者ニーズの変化のスピード
変化の速いネットビジネスなどは、変化を吸収するフェーズ（繰り返し）を当初計画に盛り込む

新システム分野への展開

PROJECT-R
Ruby X TECHNO PROJECT

Ruby業務システム開発

・初期コストが少ない・・・全てオープンソースで構築できる（DBも）
・テストが容易である・・・Rails + アジャイル ⇒ 短期繰り返し開発

非定型の業務へ

・EXCELを使った統計分析の報告業務・・・担当者の交替により継続性がなくなる
　定期的な報告時の基礎データをシステム化し、継続性のある報告（質の向上）
・エビデンスのための定期的な記録作成
　点検・ヒアリング（機械類・システム・プロジェクト・組織・企業）をシステム化し、
　時系列分析や情報の見える化が可能
・基幹の業務から外れたもの
　トラブル報告、問題点管理、派遣社員の作業時間管理、・　・　・

的確な指示・指導・改善へ
（ホワイトカラーの生産性向上）

システム提供側の変革

PROJECT-R
Ruby X TECHNO PROJECT

イノベーション　Rubyアジャイル開発を業務システムに適用

顧客創造　これまで、システム化の対象範囲から外れていたものを
システム化の新分野として、新しい顧客を創り出してくれる。

ビジネス拡大

オープンソースの活用技術（リテラシィ）
アジャイルによる業務システム開発
新システム分野へのチャレンジ

Rubyの広がり

PROJECT-R
Ruby X TECHNO PROJECT

①米国で認められた製品の逆輸入（海外で認知）

②競争激化によるコストダウン要求

有償ソフトから無償あるいは低価格ソフトへ

OSから言語まで、すべてのミドル製品がオープンソースで揃う

③国の支援（世界で通用する国産技術）

経済産業省（IPA）や総務省の支援

国・地方自治体での調達指針

2006年総務省ガイドライン:オープンな標準製品

⇒ **2011**年**3**月 **JIS**化 完了

④C/Sシステムを**Web**化するタイミング ⇒ クラウドサービスの時代

⑤ Ruby/Railsの業務システム開発標準

Rubyで新たな価値の創造を

Ruby × Techno Project

ご清聴ありがとうございました

笑顔を結ぶベストパートナー テクノプロジェクト

TPJ TECHNO PROJECT