SPI Japan 2014
2014/10/15

HITACHI
Inspire the Next

Rubyを使った開発プロジェクトでの取り組み

株式会社　日立ソリューションズ
技術開発本部　Rubyセンタ
細美　彰宏

HITACHI
Inspire the Next

Contents

HITACHI
Inspire the Next

1. Rubyの紹介

HITACHI
Inspire the Next

❶ Ruby：日本発のオブジェクト指向スクリプト言語

- **まつもとゆきひろ氏が開発し、公開**（1993年～）
  - コミュニティによる開発のオープンソース言語
  - BSDLとRubyライセンスのデュアルライセンス
- **特徴**
  - オブジェクト指向、動的型付け、インタプリタ型
  - シンプルな文法、高い生産性と保守性
  - Javaに比べて、ソースコードを短く記述可能
- **標準化活動**
  - 2011年3月、JIS規格（JIS X 3017）制定
  - 2012年4月、国際規格化（ISO/IEC 30170）

HITACHI
Inspire the Next

❷ Ruby on Rails：Webアプリケーションフレームワーク

- David Heinemeier Hansson氏が開発し公開（2004年～）
  - フルスタックのWebアプリケーションフレームワーク
  - MITライセンス
- 動画「Creating a weblog in 15 minutes」で注目
  - 高い生産性
- 基本理念
  - 同じことを繰り返さない（DRY：Don't Repeat Yourself）
  - 設定より規約（CoC：Convention over Configuration）
- 利用事例
  - 海外：Basecamp、Twitter、Hulu、GitHub、など
  - 国内：クックパッド、食べログ、など

HITACHI
Inspire the Next

2. 日立ソリューションズの取り組み

2-1 日立ソリューションズの取り組み

HITACHI
Inspire the Next

❶ 早い時期から組織的にアクション

Ruby開発の改善活動と、
大規模とアジャイル型
開発の2事例を紹介

- 大規模のシステム開発にも
  Ruby適用を挑戦
- 中小規模の開発を中心に促進
  実践を通して、実績とノウハウを蓄積

2009年

- Ruby普及促進を目的に
  専門組織Rubyセンタを設立(12月)

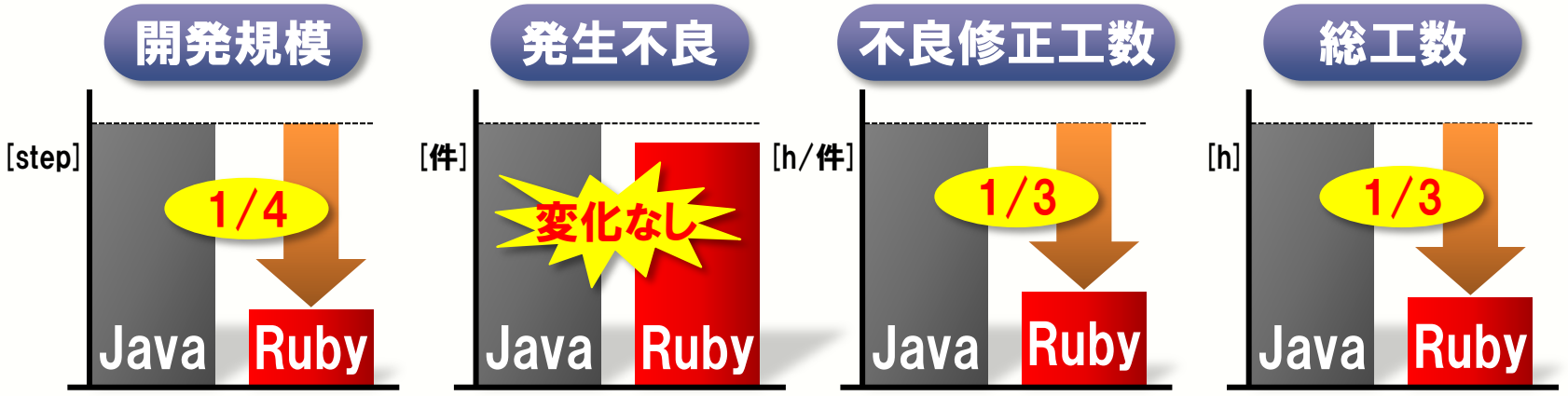
2008年

- 社内システムに、Ruby on Railsを適用して、
  生産性や拡張性を評価(4月～)
- Ruby発祥の地、島根県松江市に事務所開設(10月)

HITACHI
Inspire the Next

② 業務システムをRubyで実装して評価（2008年）

- 業務システムの一部（4画面）をJavaとRubyで実装して比較
- 仕様書、テスト項目は同じものを使用
- 開発規模が大幅に削減できることを確認



※ 総工数はプログラミングと単体テスト工程のみ

HITACHI
Inspire the Next

❸ 専門組織 Rubyセンタ

- Ruby/Railsを使ったシステム開発に対応する専門組織
  - 顧客提案、見積支援、技術問合せ、技術者育成
- 開発技術の整備
  - 開発ノウハウの蓄積、開発ガイドの整備
  - 共通ライブラリやツール開発、開発環境の構築
- Ruby/Railsのサポートサービス
  - 問題解決支援
  - セキュリティ情報配信、新規リリースの情報提供
- Ruby推進団体との協調

HITACHI
Inspire the Next

3. Ruby開発の課題と改善

3-1 Ruby開発の課題と改善

HITACHI
Inspire the Next

❶ 大規模適用への課題

- 大規模システムのRuby開発実績が少なく、未整備
  - プロジェクト個別に開発方法、環境を用意
  - 多人数でのRuby開発方法が未整備
- Ruby技術者不足
  - Java技術者に比べると、圧倒的に少数
  - 多人数の開発体制の構築が困難

❷ 大規模適用のための改善策

- 大規模システムのRuby開発手法と開発環境の整備
  - 過去の開発実績でのノウハウをベースに整備
  - 比較的新しい技術やツールの導入を方針に
- Ruby経験の浅い要員のために、開発作業ガイドで標準化

HITACHI
Inspire the Next

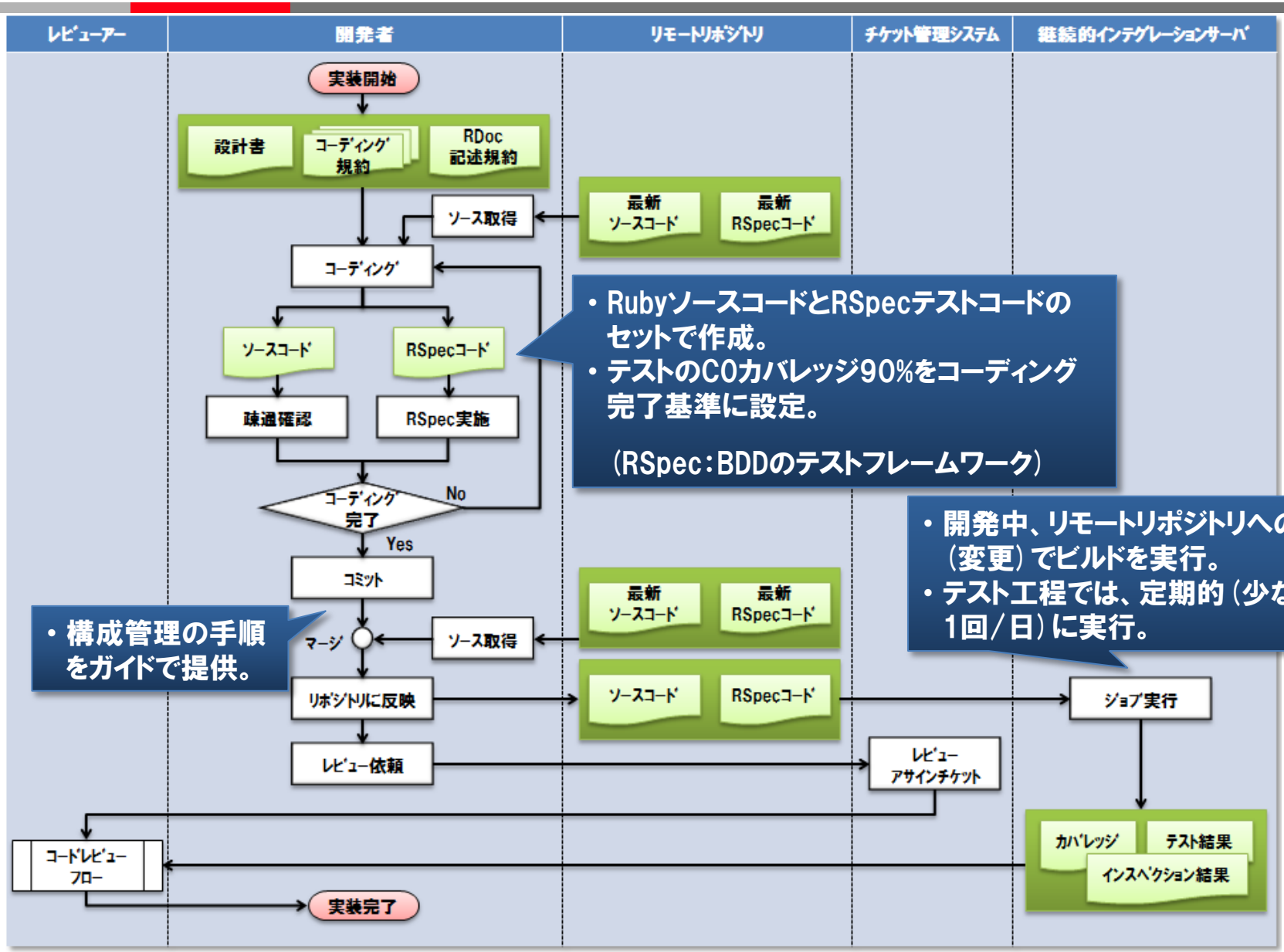
❸ Ruby開発手順

- プロジェクトの立ち上げから設計、実装、テストまでの手順
- 多人数で開発するため、レビューをルール化
- テスト自動化、継続的インテグレーションの適用を推奨

#	主な文書	概要
1	開発ガイド	設計、実装、テストでの作業手順とルール、等
2	設計留意事項	Railsの特性に合った設計のための留意事項
3	コーディング規約	Ruby/Rails命名規約、実装規約や推奨事項
4	品質管理ガイド	Ruby/Railsを使った開発における品質管理
5	性能対策ガイド	性能問題発生時の解析方法と対策方法
6	適用チェックリスト	Ruby/Rails適用可否チェックリスト
7	体制構築ガイド	推奨する開発体制と必要な技術スキル

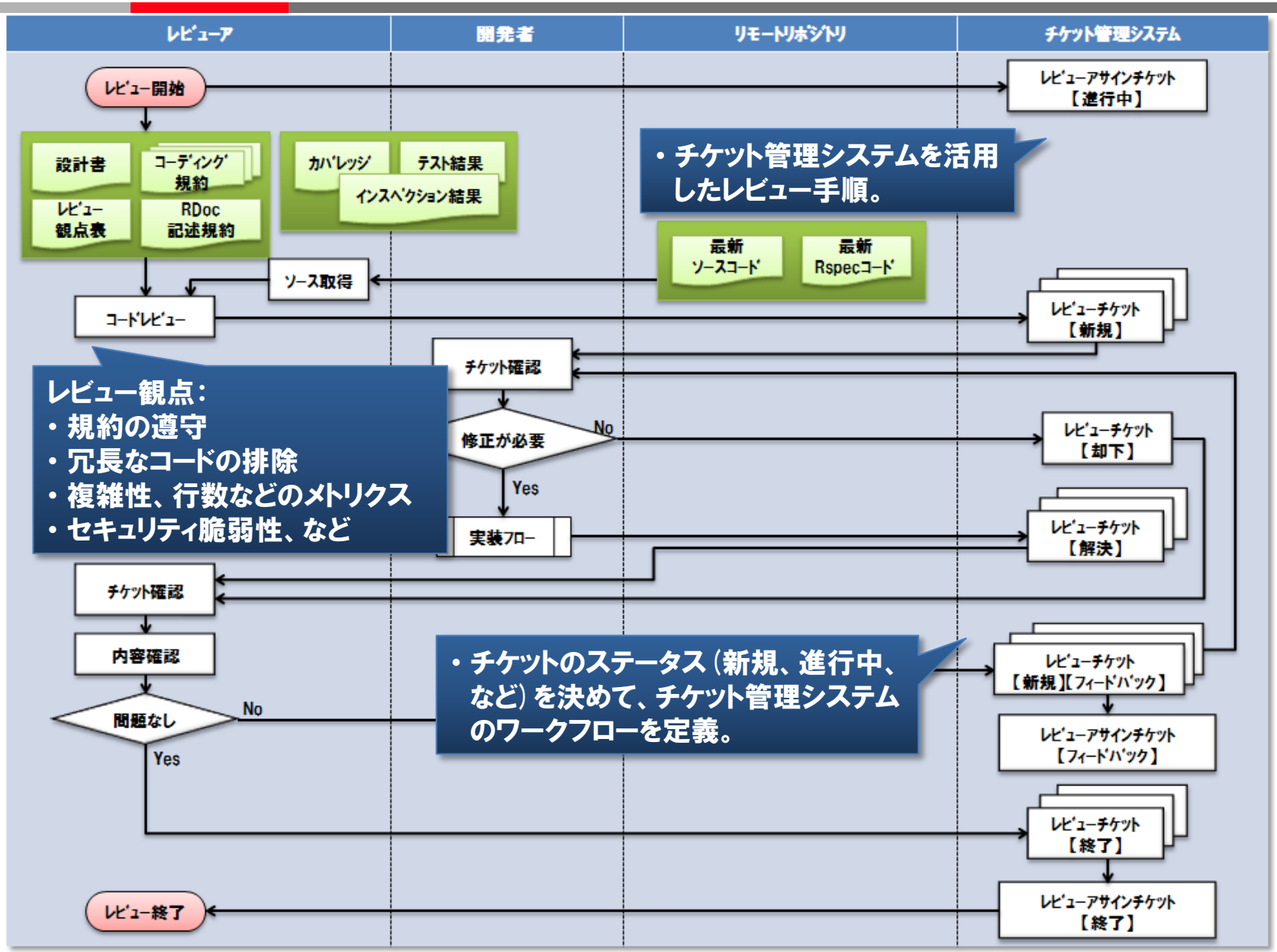
3-3 Ruby開発手順： 実装手順

HITACHI
Inspire the Next



3-4 Ruby開発手順： コードレビュー

HITACHI
Inspire the Next



3-5 Ruby開発環境

HITACHI
Inspire the Next

OSSを活用、すぐに使える状態で提供

＜開発者が共通に利用する環境＞

継続的インテグレーション
サーバ：Jenkins

・自動テスト、静的解析、ステップ数計測を日々実行して、レポート

定期的にソースを取得

バージョン管理システム
Git、Subversion

関連付け、更新の連携

・タスク、バグのトラッキングと情報共有（Wiki）

チケット管理システム
Trac、Redmine

＜開発者のPCに構築する作業環境＞

仮想マシン
（Rubyアプリの動作確認環境）
- CentOS
- Ruby
- Ruby on Rails
- Git、Subversion
- PostgreSQL

・実装、疎通確認
・コードレビュー
・自動テスト、静的解析の結果確認
・チケットの登録、確認等

チェック機能や文字列変換等、日本の企業システムで使用頻度の高い部品群を提供

HITACHI
Inspire the Next

4. 適用事例

HITACHI
Inspire the Next

❶ 大規模Webシステム開発

- 行政関連のデータを一元管理して、効率的に運営するための業務システムをRubyで再構築

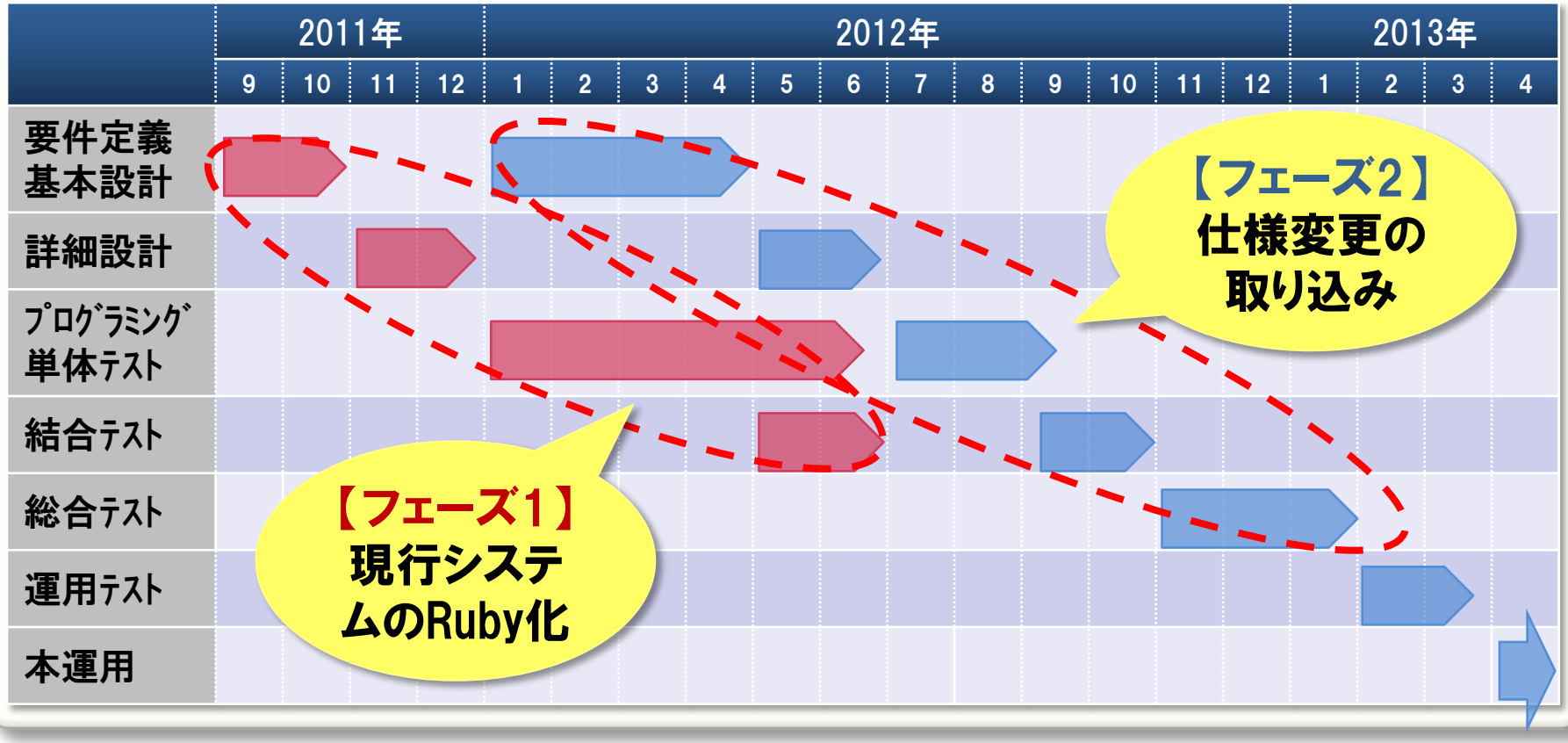
- 機能追加、制度改正時の対応効率化を目的にRuby化
  - Java/C言語で構築した現行システムは、度重なる改造と機能追加で複雑化
  - 保守性向上を狙って、Rubyへ移行

- 開発期間
  2011/9～2013/4

HITACHI
Inspire the Next

❷ 開発スケジュール

- Rubyの特徴である高い開発効率と保守性を活かして、2フェーズに分けて開発



HITACHI
Inspire the Next

❸ 開発規模

- 開発規模を現行システムの約1/5に削減
  - ただし、テストコードを含まない

#	項目	現行システム(参考)	開発対象
1	開発言語	オンライン機能：Java バッチ機能：C言語	Ruby、Ruby on Rails
2	画面数	270画面	300画面
3	帳票数	60帳票	60帳票
4	バッチ処理数	30本	30本
5	テーブル数	90テーブル	100テーブル
6	ステップ規模	約500Kstep	約100Kstep

HITACHI
Inspire the Next

④ 品質安定化

- **テスト方針**
  - テスト自動化を導入、日々、チェックして、品質を確保
  - テストコード作成基準C0カバレッジ100%
- **テスト自動化とリファクタリングによる品質の安定化**
  - デグレードの早期検出と対策による品質維持・向上
  - Ruby/Railsとテスト自動化によって、
    変更の影響範囲特定が容易に（コードの変更が怖くない）
- **カバレッジ100%達成重視により、テストコードが肥大化**
  - テスト作成基準の見直し
    （100%→90%）

4-5 適用事例2

HITACHI
Inspire the Next

❶ アジャイル型開発

- 教務関連のデータをExcelで管理していたが、対象範囲が広がり、管理業務が困難になり、システム化
- 早くリリースしたい、ニーズを具体化しながら、徐々に機能追加したいという要望があり、有効性検証も含め、アジャイルを適用
  - 開発途中で動くシステムを見せ、ユーザに仕様確認
  - アジャイルとRubyの親和性の検証
- 開発期間：2010/9～2011/2
  - 1ヶ月のスプリントを3回実施+1回予備

<計画>

項目	1	2	3	計
ストーリー	7	7	7	21
タスク	48	32	49	129

<実績>

項目	1	2	3	計
ストーリー	5	4	7	16
タスク	35	22	49	106

4-6 適用事例2： アジャイル型開発

HITACHI
Inspire the Next

❷ 開発の概要

ビジネス要件

項目	1	2	3	計
ストーリー	7	7	7	21
タスク	48	32	49	129

スプリント計画
- 目標決定
- 実現する ストーリー選択

スプリント
- 日次スクラムミーティング
- バーンダウンチャート

デモを実施して、フィードバック

実行可能なシステム（23画面、6帳票）

- ふりかえりミーティング

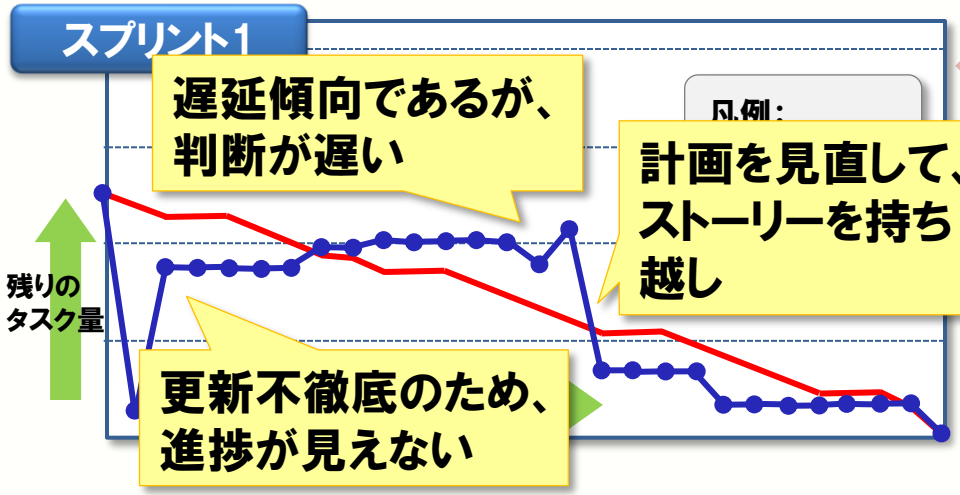
プロダクトバックログ
- システムを実現するために必要な全ての機能・成果物の一覧
- プロダクトオーナーによって、実施有無と優先順位付け

スプリントバックログ
- スプリントで行うストーリーの一覧
- タスクに分解し、各タスクの作業時間を見積もる

4-7 適用事例2： アジャイル型開発

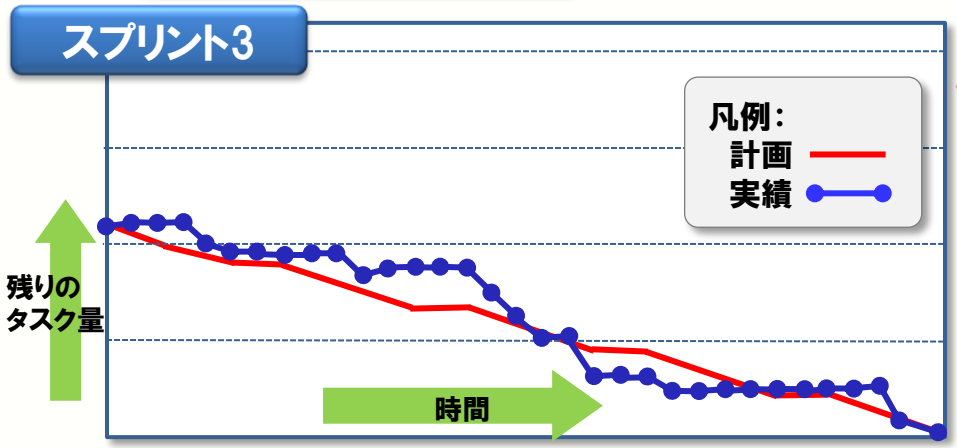
HITACHI
Inspire the Next

❸ バーンダウンチャートの傾向



正確な進捗管理ができていない

- 開始時、チケット管理システムに障害発生
- タスク分解の粒度にバラつき
- タスク状態の日次更新が不徹底



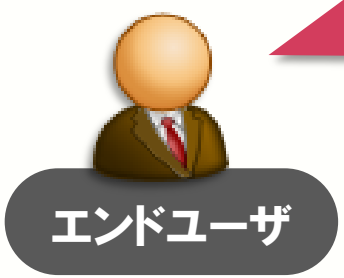
チームの習熟度が上がり、開発がスムースに進行

- 適切なタスク分割、優先度付けにより、進捗管理ができるように
- 1タスクは1日で完了する作業を基準に設定

4-8 適用事例2： アジャイル型開発

HITACHI
Inspire the Next

❹ プロジェクトメンバのコメント



ユーザニーズの掘り起こしに、
今回の手法は有効。
漠然とした要求から、形に
なったことは評価できる。

早い段階で、動くシステムからフィードバック
が得られ、ユーザ要件を掘り起こしやすい。
反面、要望・仕様変更が多発するため、
バックログの管理が大切。

変更や追加要望に対して柔軟な対応が期待
できる点で、Rubyとアジャイルは親和性が高い。

HITACHI
Inspire the Next

5. まとめ

5-1 まとめ

HITACHI
Inspire the Next

❶ 大規模向けRuby開発手法と開発環境の整備

- Ruby開発手法と開発環境を整備して、適用推進
  - 大規模システムとアジャイル型開発
- 今後も継続整備
  - 適用プロジェクトからフィードバックを受けて改善

❷ 今後の課題

- Rubyおよび大規模開発、アジャイルの両方を理解する技術者の育成
- Ruby＋大規模開発＋アジャイル型開発の開発手法の検討

HITACHI
Inspire the Next

END

Rubyを使った開発プロジェクトでの取り組み

株式会社　日立ソリューションズ
技術開発本部　Rubyセンタ
細美　彰宏