

平成22年度しまねIT産業振興事業

Ruby ビジネスモデル研究実証事業  
報告書

作成日：2011年3月3日

作成元 株式会社プロビズモ

## 報告にあたって

貴県益々ご清栄のこととお慶び申し上げます。この度は、「Ruby ビジネスモデル研究実証事業」に弊社からご提案を採択賜り、厚く御礼申し上げます。

この度、計画しておりました本事業が完了致しましたので、本報告書にて本事業の内容をご報告申し上げます。

本報告書が、島根県並びに島根県内ソフト IT 企業様のお役に立てることを切に願う次第であります。

何卒、ご査収の程、よろしくお願い申し上げます。

株式会社プロビズモ

代表取締役社長

鎌田 大輔

# 目次

1 はじめに .....	- 1 -
2 本事業における開発実証の目的と概要 .....	- 1 -
2-1 実証事業の背景と目的 .....	- 1 -
2-2 実証事業の対象システム .....	- 1 -
2-3 実証体制（ユーザ含む） .....	- 2 -
2-4 実証スケジュール .....	- 4 -
3 準備内容 .....	- 5 -
4 実証する RUBY の特徴を活かす開発手法 .....	- 5 -
4-1 実証する開発手法、プラクティスの選定理由 .....	- 6 -
4-2 実証する開発手法、プラクティスについて .....	- 6 -
4-3 実証した開発手法、プラクティスの有効性と課題 .....	- 12 -
4-3-1 システム規模 .....	- 12 -
4-3-2 生産性 .....	- 13 -
4-3-3 ソースコード .....	- 14 -
4-3-4 計画要求数/受入要求数/実現要求数 .....	- 14 -
4-3-5 割込(変更)の受入要求数/実現要求数 .....	- 14 -
4-3-6 有効性と課題 .....	- 15 -
4-4 実施体制の有効性と課題 .....	- 15 -
4-4-1 会議の実施結果 .....	- 15 -
4-4-2 コストについて .....	- 16 -
4-4-3 有効性と課題 .....	- 17 -
4-5 ユーザ満足度に及ぼす有効性と課題 .....	- 17 -
4-5-1 開発したシステムに対する評価 .....	- 17 -
4-5-2 スケジュールと納期に対する評価 .....	- 17 -
4-5-3 ドキュメントに対する評価 .....	- 19 -
4-5-4 イテレーション（スプリント）に対する評価 .....	- 19 -
4-5-5 コミュニケーションに対する評価 .....	- 20 -
4-5-6 元々の課題に対する評価 .....	- 20 -
4-5-7 コストパフォーマンスに対する評価 .....	- 21 -
4-5-8 有効性と課題 .....	- 21 -
4-6 品質・信頼性及び開発管理に及ぼす有効性と課題 .....	- 22 -
4-6-1 品質と信頼性 .....	- 22 -
4-6-2 分散開発環境を使用することの影響 .....	- 22 -
4-6-3 品質・信頼性の有効性と課題 .....	- 23 -
4-6-4 開発管理 .....	- 23 -
4-6-5 開発管理の有効性と課題 .....	- 28 -
4-7 開発側参加者の意見 .....	- 28 -
4-8 ユーザ側参加者の意見 .....	- 29 -

5 総括 .....	- 30 -
5-1 RUBYの特徴を活かす開発手法の今後のビジネス利用の可能性と活用分野 .....	- 30 -
5-2 RUBYの特徴を活かす開発手法の今後のビジネス利用での課題と克服策 .....	- 30 -
5-3 今回開発システムの今後のビジネス展開の可能性課題と克服策 .....	- 31 -
5-4 ビジネス利用での見積・契約での提言 .....	- 31 -
5-5 人材育成に関する提言 .....	- 31 -

## 1 はじめに

本報告書は、平成22年度しまねIT産業振興事業である『Ruby ビジネスモデル研究実証事業』における活動報告を「実施計画書 4.1.1 研究実証成果報告書」記載の構成で作成したものである。

## 2 本事業における開発実証の目的と概要

### 2-1 実証事業の背景と目的

当社は、Ruby を用いて業務アプリケーションを開発するビジネスを推進している。その中で、従来のようにウォーターフォール型の開発手法を用いると、すべての機能を開発するまで本番運用ができず、稼動開始までに長期間を要する。さらに、ユーザはシステムが完成するまで実際に使用することができず、ニーズとかけ離れたシステムになる危険性がある。このため、短期間で機能を実装できる Ruby の特徴を活かすことができない。これに対し、スモールスタートで早期に本番運用を開始し、徐々に拡張していくインクリメンタル型の開発ができれば、早期に投資の回収が可能となり、システム開発の投資効率が改善できる。また、ユーザニーズを吸い上げながら徐々に拡張することによって、ユーザにとって価値のある機能を作りこむことができると考える。

当社は、オブジェクト指向、動的型付け、インタプリタ型及びシンプルな文法などの Ruby の特徴を活用すれば、短期間でシステムをサービスすることができ、かつ、システム拡張に要するコストの肥大化を抑えたインクリメンタル開発ができると考える。

ところが、現状の請負契約では、システムを納入、検収後でなければ、本番運用を開始できない。このため、スモールスタートで頻りにリリースを行うには、その都度、納入、検収を行わなければならない、インクリメンタル開発の適用を困難なものとしている。

そこで本事業では、ユーザと開発チームを同じチームとし、Ruby を活用したイテレーション単位での本番運用を前提としたインクリメンタル開発を実施して、その有効性を実証する。

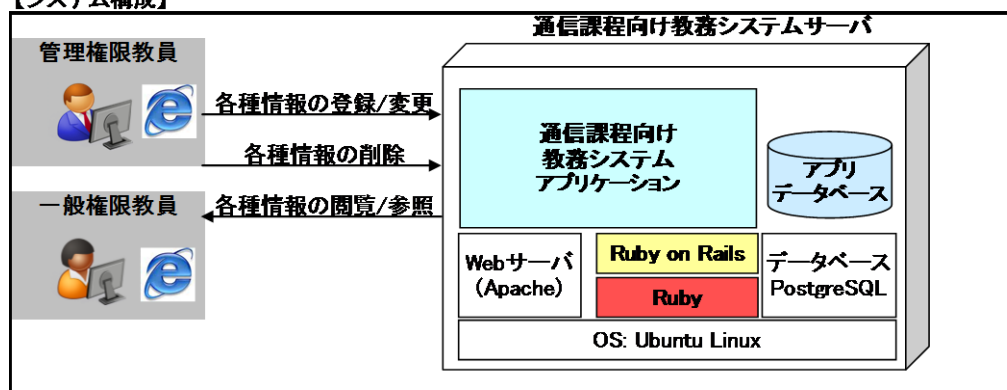
適用対象システムとして、益田永島学園明誠高等学校向けの通信課程向け教務システムを開発する。

### 2-2 実証事業の対象システム

本事業の適用対象システム「益田永島学園明誠高等学校向けの通信課程向け教務システム」は、一般的な全日制のそれとは異なり、生徒毎に異なるカリキュラム(履修計画)を作成し、スクーリングやレポートの結果より成績を導き出す必要がある。また、出欠の概念も無く、従来の全日制教務システムの拡張としての構築は困難である。

【システム構成】Web アプリケーションで構成、概要は以下の通りである。

## 【システム構成】



【生徒管理】生徒の個人情報(転学前/在籍)を管理する。

【履修管理】卒業までの履修計画/実績情報を管理する。

【成績管理】成績情報(試験/評定/単位/スクーリング/レポート)を管理する。

指導要録/調査書作成に必要な情報を管理する。

【システム管理】学校/科目/教材マスタ、システム利用ユーザを管理する。

【検索】生徒の個人情報/成績情報/履修計画/履修実績情報を多角的に検索する。

【出力帳票】①成績通知表②生徒指導要録③学習(履修)計画書④費用請求書

⑤履修科目確認票⑥使用教材一覧表⑦単位認定試験得点一覧表⑧その他各種一覧表

### 2-3 実証体制(ユーザ含む)

本事業の実証体制は、以下の通りである。

開発チームは、株式会社プロビズモ(以下、プロビズモ)、株式会社日立ソリューションズ(以下、HISOL)、益田永島学園明誠高等学校(以下、明誠高校)の3機関により構成するものとし、それぞれの役割を下記に記述する。

※日立ソリューションズ株式会社は、2010年10月に日立ソフトウェアエンジニアリングと日立システムアンドサービスの合併により発足。

#### [プロビズモ]

プロビズモは、主に、システムの実装チームとして機能する。メンバは、プロビズモのRuby開発実績(Ruby開発経験1年)を持った技術者により構成される。

また、プロジェクトの窓口としての役割を持つ。

実装チーム(開発者)とユーザ窓口は、出雲本社を中心に活動する。プロジェクトのリーダーは、東京支社を中心に活動する。

#### [HISOL]

Rubyによるインクリメンタル開発の実績が豊富なHISOLは、スクラムマスターとして、開発手法のコンサルティングやRuby技術支援を行い、プロジェクトを円滑に推進する為のサポートを行う。

また、分散開発を実現する為の環境の提供及び、そのサポートを行う。

活動の中心は、HISOL本社がある品川になるが、コンサルタント主担当は実装チームやユーザへの支援の為に、HISOL松江事業所で活動することもある。

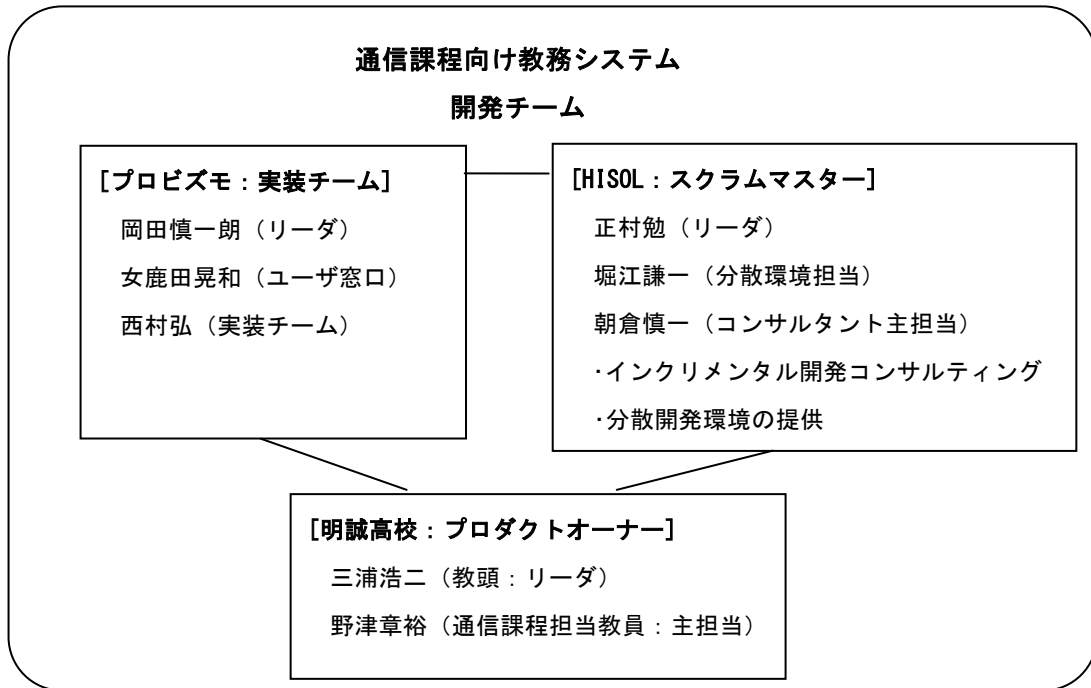
#### [明誠高校]

明誠高校は、プロダクトオーナーとして、実装する初回要求機能の抽出及び、優先度付けを行う。

また、分散開発環境を使用して各イテレーションで実装したシステムの動作確認を行い、次回

イテレーション要求としての機能追加、変更及び優先度付けを行う。

益田明誠高校内を中心に活動する。



## 2-4 実証スケジュール

本事業の委託業務期間は、以下の通りである。

始期：平成 22 年 9 月 9 日

終期：平成 23 年 2 月 10 日

本事業のスケジュールの計画と実績は、以下の通りである。

### ◎計画

作業項目	主担当	2010年				2011年	
		9月	10月	11月	12月	1月	2月
計画	明誠・スクラムM・実装T	➤					
本番運用	明誠		➤	➤	➤	➤	➤
課題抽出・フィードバック	明誠・スクラムM・実装T		➤	➤	➤		
イテレーション1	実装T		➤				
└ ユーザ管理	実装T		➤				
└ 基本情報管理	実装T		➤				
└ 保護者情報管理	実装T		➤				
└ 転学校成績管理	実装T		➤				
イテレーション2	実装T			➤			
イテレーション3	実装T				➤		
イテレーション予備	実装T					➤	
最終評価	明誠・スクラムM・実装T					➤	

### ◎実績

作業項目	主担当	2010年				2011年	
		9月	10月	11月	12月	1月	2月
計画	明誠・スクラムM・実装T	➤					
本番運用	明誠		➤	➤	➤	➤	➤
課題抽出・フィードバック	明誠・スクラムM・実装T		➤	➤	➤		
イテレーション1	実装T		➤				
└ ユーザ管理	実装T		➤				
└ 基本情報管理	実装T		➤				
└ 保護者情報管理	実装T		➤				
└ 転学校成績管理	実装T		➤				
イテレーション2	実装T			➤			
└ 成績管理	実装T			➤			
└ 履修管理	実装T			➤			
└ 帳票出力	実装T			➤			
イテレーション3	実装T				➤		
└ 成績管理	実装T				➤		
└ 請求管理	実装T				➤		
イテレーション予備	実装T					➤	
└ システム管理	実装T					➤	
最終評価	明誠・スクラムM・実装T					➤	

### 3 準備内容

#### (1) 新規事業についてのヒアリング

益田明誠高校理事長から本事業で開発するシステムを利用する、通信課程事業についての新規事業部分のヒアリングを実施した。新規事業部分とは、全国の塾等をサテライト校として、生徒の広域募集を行うというもので、本ヒアリングの結果をもとに本事業で選択する開発方法を決定した。プロジェクト開始時点では、全ての要件が確定しないことと早期事業立上げが明確になった。

実施日：2010年8月26日。

#### (2) キックオフミーティング

プロジェクト開始にあたり、開発チーム参加者並びに益田明誠高校理事長、プロビズモ社長、副社長も出席し、益田明誠高校校長室とプロビズモ東京支社とを本事業で使用するTV会議システムを利用し、2拠点で同時にプロジェクト計画を相互確認した。尚、プロビズモ副社長、実装チームのリーダー以外のメンバとスクラムマスターコンサルタント主担当は、益田明誠高校校長室で参加した。このキックオフミーティングでプロダクトオーナーリーダーと主担当者へ本事業で実施する開発手法の説明を実施した。

実施日：2010年9月9日

#### (3) イテレーションの試行

インクリメンタル開発の学習目的で、イテレーションの試行を実施した。想定した機能（今回開発したシステムで横断的に利用する機能：ユーザ管理機能・ログイン機能・データベースアクセス機能・帳票出力機能）を元にプロトタイプ開発を実施し、技術的課題（Ruby on Railsの機能及びモデル関連・帳票出力機能・和暦対応）の抽出および分散環境上の各種ツールの使い方を学習した。

実施期間：2010年9月10日～2010年9月22日

#### (4) 実装チームの業務理解

通信課程学校業務を実装チームが理解する為に、益田明誠高校にて、プロダクトオーナーのリーダーと主担当者から実装チームのユーザ窓口と実装担当に対して、通信過程学校業務を説明した。

実施日：2010年10月5日

#### (5) 分散開発環境のプロダクトオーナー主担当者への説明

本事業で使用する分散環境上の各種ツールの使い方を益田明誠高校通信課程担当室で、実装チームの実装担当から説明した。

実施日：2010年10月28日

### 4 実証するRubyの特徴を活かす開発手法

本事業で開発するシステムは、新規事業向けのシステムであり、すべての要件、機能が未確定でも、早期にシステムを導入し、ビジネスを開始する必要性があった。ユーザが新規ビジネスを行う上で以下のよ様な課題を抱えていた。

- ・ 早期にシステムを導入し、事業を軌道にのせる
- ・ 業務にマッチした機能を有するシステムの導入
- ・ 事業拡大に合わせて拡張できる柔軟なシステムの導入

また、ユーザは、システム開発に携わった経験がなく、要件定義が難しい状況にあった。それに対し、現在、多くのシステム開発で採用されているウォーターフォール型の開発手法を採用した場合、以下のよ様な問題があり、ユーザが抱える課題を解決できない。

- (1) すべての機能を開発するまで本番運用ができず、稼働開始までに長期間を要する。そのため、ビジ

ネスチャンス逃す恐れがある。

- (2) すべての仕様を決定してから開発を行うため、開発途中で判明する本来のニーズや、市場の変化に柔軟に対応しなければならない新規事業などに対応できない。
- (3) ユーザはシステムが完成するまで実際に使用することができず、業務にマッチしないシステムになる危険性がある。

そこで、迅速かつ適応性が高いシステム開発を行う必要があると考えた。

#### 4-1 実証する開発手法、プラクティスの選定理由

実証する開発手法は、一ヶ月単位の短期開発を複数回実施してシステムを作り上げるインクリメンタル開発である。インクリメンタル開発では、一ヶ月の開発終了ごとにシステムの本番リリースを行い、運用の中で発生するユーザからのフィードバックを反映しながら、徐々に拡張を行っていく手法である。本番リリースは、ユーザが実際に業務でシステムを使うことを示す。インクリメンタル開発の特長を以下に示す。

- (1) 短期間でシステムをサービス可能
- (2) スモールスタートで早期に運用を開始することにより、システム開発の投資効率の改善が可能
- (3) ユーザニーズを吸い上げながら徐々に拡張することによって、ユーザにとって価値のある機能を作りこむことが可能

さらに、少ない記述量でシステムを開発できる Ruby を適用すれば、以下が実現できる。

- (1) 短期間で機能を作り込むことができ、より多くの要望を実現可能
- (2) 開発途中で判明する本来のニーズや市場の変化による仕様変更に対しても、短期間で適応可能
- (3) 開発途中の変更要求も含め、機能を短期間で実装できるため、システム拡張に要するコストの肥大化を抑えることが可能

Ruby を使ったインクリメンタル開発を行うことにより、ユーザの抱える課題を解決できると考え、この手法を採用した。

また、今回は、ユーザ、開発者、アジャイル開発支援者が地理的に離れた拠点に分散しており、同じ場所で作業することが困難であった。そこで各拠点をネットワークで結んだ開発環境を用いた分散開発を行い、その有効性も併せて検証した。

#### 4-2 実証する開発手法、プラクティスについて

インクリメンタル開発は、図 4-1 に示すように1ヵ月単位の実装を複数回実施し、インクリメンタルに開発を行った。計画時に要件を整理して優先度を決定。これを元にイテレーションで実装を行った。各イテレーション終了時には、実際に動作するものをユーザに提供した。評価では、本事業全体の評価を実施した。

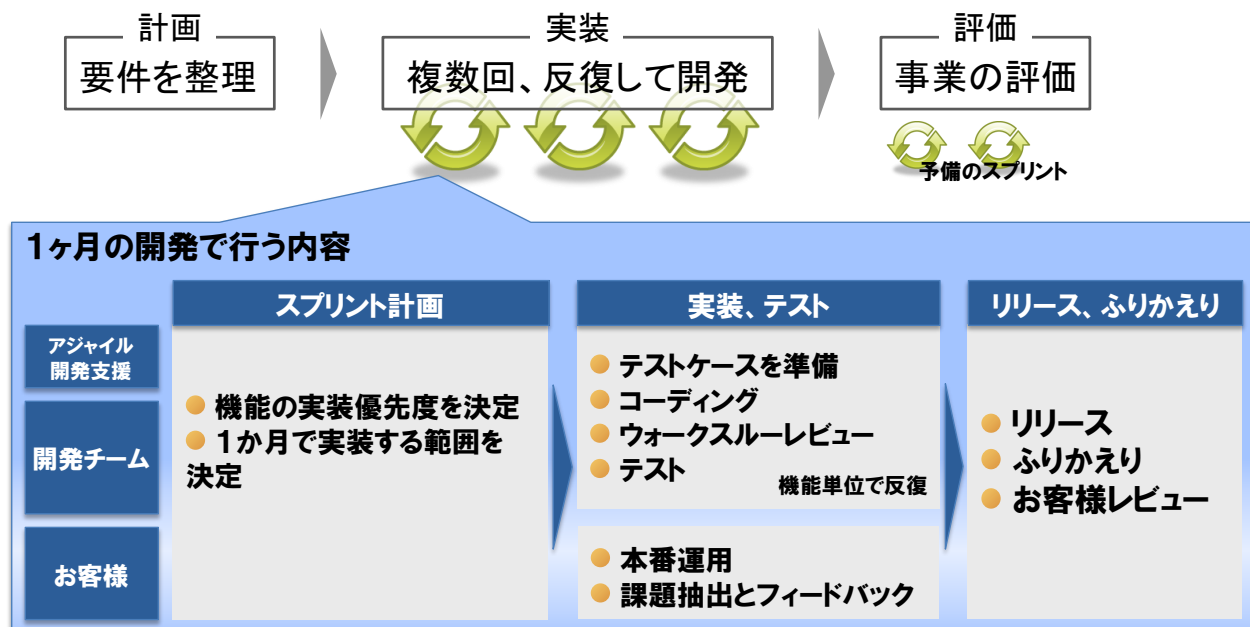


図 4-1 インクリメンタル開発概要

(1) 分散開発環境

本事業では、地理的に離れた複数拠点をセキュアなネットワークで結んだ仮想的な共有環境を使用した。分散開発環境には、構成管理、懸案管理、テスト自動化などの開発ツールと、ユーザも利用できる実行環境がある。また、Web会議といったコミュニケーションツールも提供し、チームメンバが一体となって開発を実施した。図 4-2 に分散開発環境の構成図を示す。

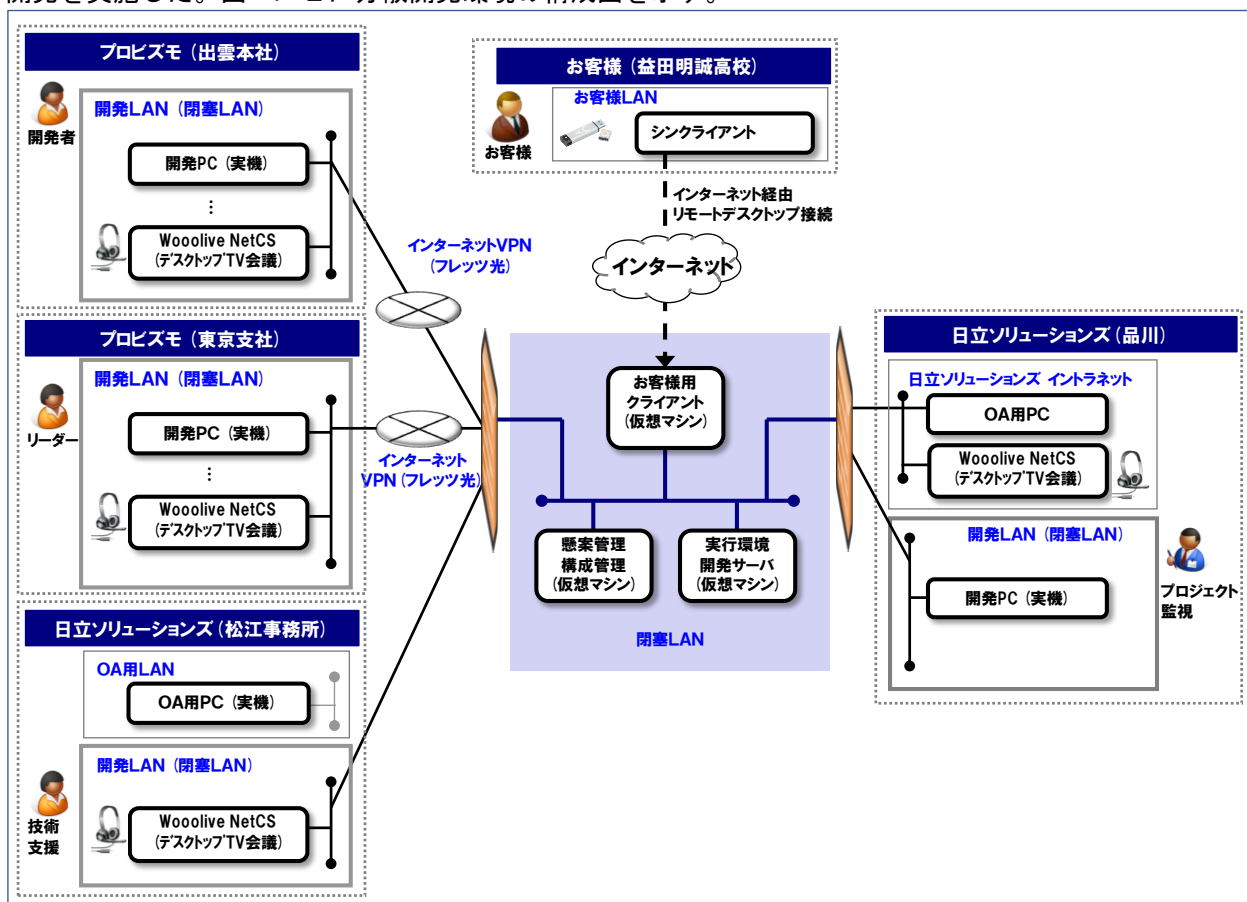


図 4-2 分散環境構成図

(2) 開発におけるプラクティス

(a) スプリント計画

ユーザを含めた開発チーム全員で、バックログの優先度を見直し、スプリントの目標を決定した。その後、要求を実現するためのタスクを列挙し、スプリントバックログを作成した。

(b) 30日のイテレーション

一ヶ月を開発の単位として、3回繰り返してシステムを開発した。

(c) ウォークスルーレビュー

ペアプログラミングの代用としてソースコードのウォークスルーレビューを実施し、不良の早期発見、ソースコードの洗練、知識の共有を行った。

(d) テスト駆動開発

テスト駆動型を意識した開発を実施した。テストコードは書かず、開発前にテストシナリオとテストケースを文章で定義し、その定義に沿った開発を行った。

スプリント2のみ、RSpecを使用したテストコードを用意し、テスト駆動開発を試行した。

(e) テスト自動化

図4-3に示す自動テストツール「anyWarp Capture/Replay」を使用し、リリース前にテストを実施した。デグレード防止用に必要最低限のテストケースを各スプリントで実施した。このツールにより、画面操作の主なテストは自動化し、テストデータはユーザより提供されたものを使用した。

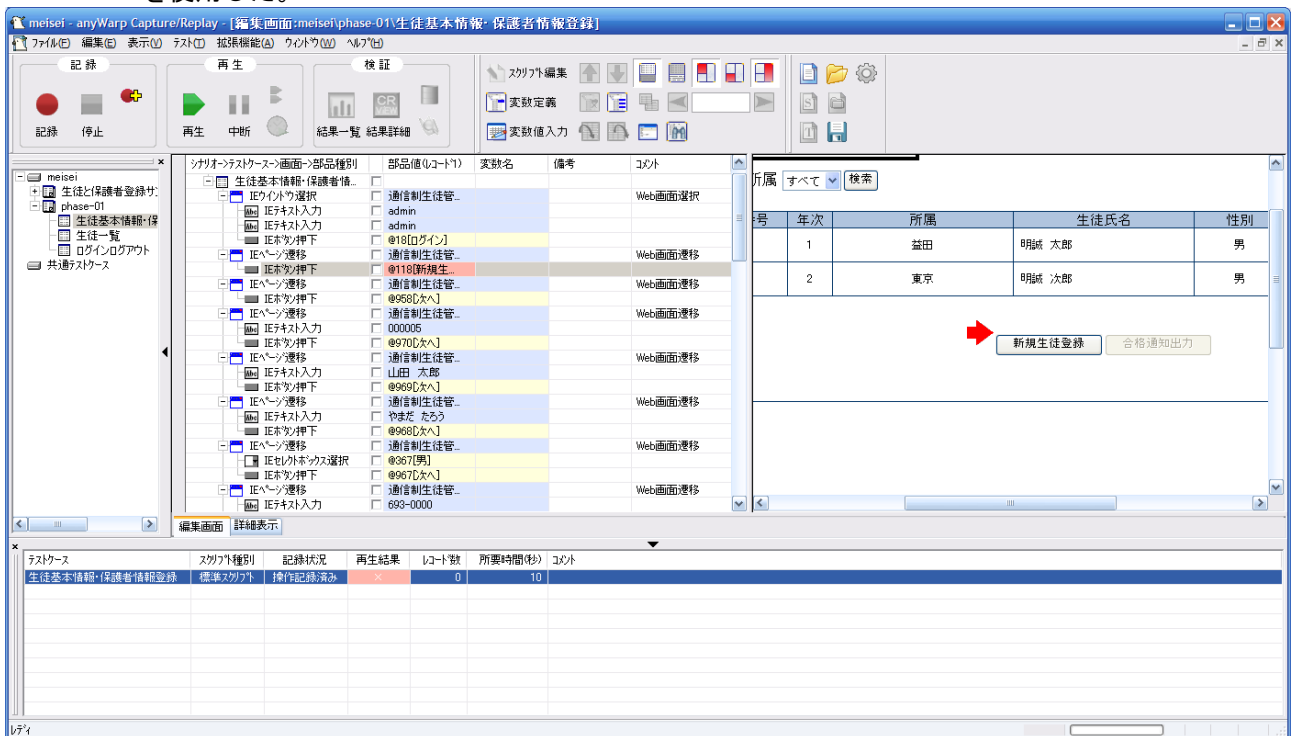


図 4-3 自動テストツール

anyWarp Capture/Replay は、画面操作の記録と再生を行う自動回帰テストツールである。図4-4に示すように、記録したテストスクリプトを蓄積することにより、次回スプリントのテストを自動化する。また、表示結果の画面スナップショットを取得し、画面の比較も自動的に行うことができる。

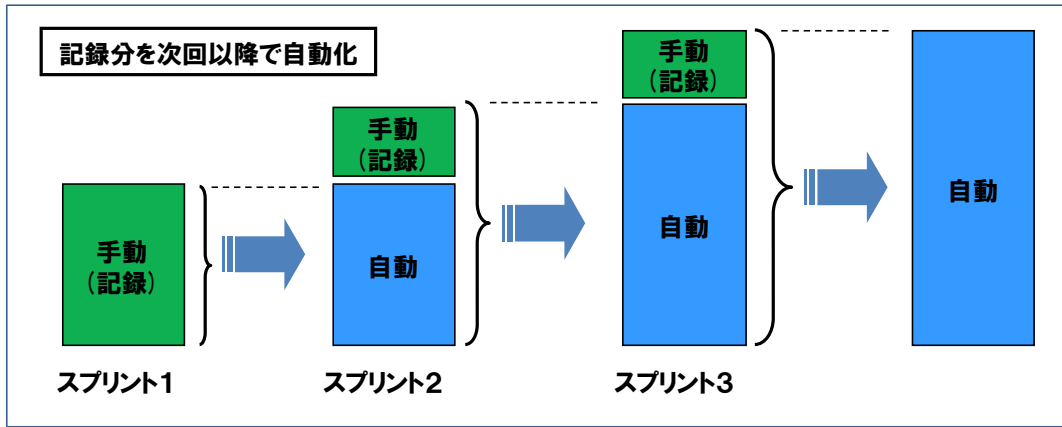


図 4-4 画面操作テストの自動化概要

(f) コーディング規約

コーディング規約を定義し、統一したコーディングを行なおうとした。コーディング規約の適用は、図 4-5 に示すコードインスペクションツールを使用して自動的にチェックを行い、ルール違反を指摘する。

<<通信制生徒管理システムに戻る

レポート日時: 01月13日 17:22    レポートID: 129490693813    総指摘件数: 272件

指摘ルール	重要度	指摘件数
RDocコメントの記述	高	111
三項演算子を使用しない	高	2
ローカル変数名が似すぎている	中	6
RDocのインデント	中	1
RDocのフォーマット	中	25
メソッド定義内のコメント	低	47
演算子の前に空白が必要	低	17
演算子の後ろに空白が必要	低	62
デバッグ用出力メソッドを使用しない	低	1

指摘内容	重要度	場所
三項演算子は使用すべきではありません。	高	/app/models/bill.rb: L.38
三項演算子は使用すべきではありません。	高	/app/models/exam.rb: L.6

```

28 end
29 end
30
31 #検索メソッド
32 def self.search(student, school_year, term)
33   if school_year.blank?
34     search_school_year = student.course_plans.first.school_year
35   else
36     search_school_year = Date.strptime(school_year, "%Y-%m-%d")
37   end
38   search_term = term.blank? ? student.course_plans.first.term : term
39   return self.first(:conditions => {:student_id => student.id, :school_year => search_school_year, :search_term => search_term})
40 end
41
42 def self.get_report_data(id)
43   sql = <<-SQL
44     SELECT bills.enrollment_fee AS enrollmentFee,
45            bills.enrollment_fee_note AS enrollmentFeeNote,
46            bills.school_fee AS schoolFee
  
```

図 4-5 コードインスペクションツール

(g) 頻繁なリファクタリング

コードインスペクションツールの指摘内容やウォークスルーレビューの結果を元にコードの見直しを実施し、リファクタリングを実施した。

(h) ソースコードの共有

Subversion を使用してソースコードの共有を行い、開発チーム全員が最新のソースコードにアクセスできるようにした。

(i) ふりかえり

スプリントの終了時に、KPT 分析を実施した。問題点の解決策や次のスプリントで新しくチャレンジすることを整理した。ふりかえりは、ユーザを除く、実装チームとスクラムマスターで

実施した。各スプリントでのふりかえり内容を表 4-1、表 4-2、表 4-3に示す。

**表 4-1 スプリント1のふりかえり**

#	項目	内容
1	Keep	<ul style="list-style-type: none"> <li>・ TV会議を使ったユーザとのミーティング</li> <li>・ HISOLによる技術サポート</li> <li>・ 開発者自身による積極的なユーザとの接触</li> <li>・ 慣れないやり方でも短期間で無事リリース</li> <li>・ ヒアリングがきちんとできて、ユーザに満足が得られた</li> <li>・ 対面とTV会議の組み合わせが上手くいった</li> <li>・ 帳票を統合して数を減らし、その分、機能実装に充てることができた</li> </ul>
2	Problem	<ul style="list-style-type: none"> <li>・ スクラムミーティングを定時にできなかった</li> <li>・ 機能単位でのタスククローズができていない</li> <li>・ tracを十分活用できなかった</li> <li>・ プロビズモとHISOL間でTV会議を活用していない</li> <li>・ スプリント開始後に技術的課題が発生し、調査に時間を費やした</li> <li>・ 明誠高校の分散環境への接続環境構築が遅れた</li> <li>・ 分散環境の一部機能がスローダウンすることがあった</li> <li>・ TV会議のルール決めがなく、分かり難いところがあった</li> </ul>
3	Try	<ul style="list-style-type: none"> <li>・ tracに懸案事項や指摘事項なども登録したい</li> <li>・ tracできちんと進捗管理したい</li> <li>・ タスクをもっと細かくし、作業とチケットをきちんと結び付けたい</li> <li>・ TDDを少しずつ取り入れる</li> <li>・ プロビズモとHISOL間でTV会議を活用する</li> <li>・ インスペクションツールの活用</li> <li>・ TV会議のやり方ガイドをまとめる</li> <li>・ 開発者自身によるモデリングを行う</li> <li>・ Hudson CIを取り入れ、日次ビルドを行いたい</li> <li>・ 次のスプリントで明誠高校を驚かせたい</li> </ul>

**表 4-2 スプリント2のふりかえり**

#	項目	内容
1	Keep	<ul style="list-style-type: none"> <li>・ 実装チームとHISOL間でTV会議によるやりとりを実施できた</li> <li>・ 現地に行かずともTV会議を使って明誠高校、プロビズモ、HISOL間の打合せができた</li> <li>・ 開発者自身によるモデリングができた</li> <li>・ 明誠高校とのコミュニケーションが良くとれた</li> <li>・ スクラムマスター、開発者の役割が軌道に乗り出した</li> <li>・ Rubyを使った帳票関連の知識が得られた</li> <li>・ 計画-実装-フィードバックの流れが形になってきた</li> </ul>
2	Problem	<ul style="list-style-type: none"> <li>・ tracにタスクの登録漏れがあった</li> <li>・ trac上で進捗管理ができていない</li> <li>・ 見積もり時間の設定が入っていなかった</li> <li>・ TDDを行った結果、実装のパフォーマンスが落ちた</li> <li>・ インスペクションツールでの指摘に対応できていない</li> <li>・ 明誠高校だけではTV会議の接続設定が難しかった</li> <li>・ Hudson CIを有効に活用できてない</li> <li>・ 工数見積もり精度が悪く、スプリント内ですべてのタスクを消化できなかった</li> <li>・ TV会議のみでのコミュニケーションに限界があり、明誠高校のリーダーとの意思疎通ができていない箇所があった</li> </ul>
3	Try	<ul style="list-style-type: none"> <li>・ スプリントにバッファを設ける</li> <li>・ 明誠高校のリーダーとのコミュニケーションを積極的に行う</li> <li>・ メンバ全員がtracの有効活用する</li> <li>・ Ruby on Railsのプラグインを作る</li> <li>・ TV会議の手法・ノウハウについて、本格的なものを作成したい</li> <li>・ 明誠高校と実装チーム間で1週間毎に進捗会議実施する</li> <li>・ 実装した機能に対するヒアリング回数を増やす</li> </ul>

	・ ふりかえりに明誠高校も参加してもらいたい
--	------------------------

表 4-3 スプリント3のふりかえり

#	項目	内容
1	Keep	<ul style="list-style-type: none"> <li>タスクの消化サイクルが身に付き、実用性のあるバーンダウンチャートとなった</li> <li>TV 会議の画面共有機能を利用して、明誠高校と円滑なコミュニケーションが取れた</li> <li>Hudson CI を活用できた</li> <li>Ruby on Rails に適したコーディングが実施できるようになった</li> <li>スプリント 1、2 で習得したノウハウにより、技術的な障害が発生しなかった</li> <li>スプリント 1、2 の経験を活かし、見積もりの精度が向上した</li> </ul>
2	Problem	<ul style="list-style-type: none"> <li>過去分のソースコードに対し、リファクタリングしきれていない</li> <li>開発環境と本番環境の違いによる問題（文字フォントの違いなど）が発生した</li> <li>分散環境に必要なソフトウェアがインストールされておらず、明誠高校側の機能確認に支障があった</li> <li>コードインスペクションがコミット後に行われるため、ソースコードの修正が後手になってしまう</li> </ul>
3	Try	<ul style="list-style-type: none"> <li>過去分のソースコードのリファクタリングをしたい</li> <li>今回の開発手法をふまえたドキュメントを整備したい</li> <li>重複したソースコードを排除したい</li> <li>データベースアクセスのチューニングをしたい</li> </ul>

(j) 日次ビルド

スプリント 2 より、図 4-6 に示す Hudson CI を使用した日次ビルドを適用した。Hudson CI に、Ruby 用のプラグインである RubyMetrics をインストールして、日次でテストコードを実施し、その成否とコードカバレッジを取得した。

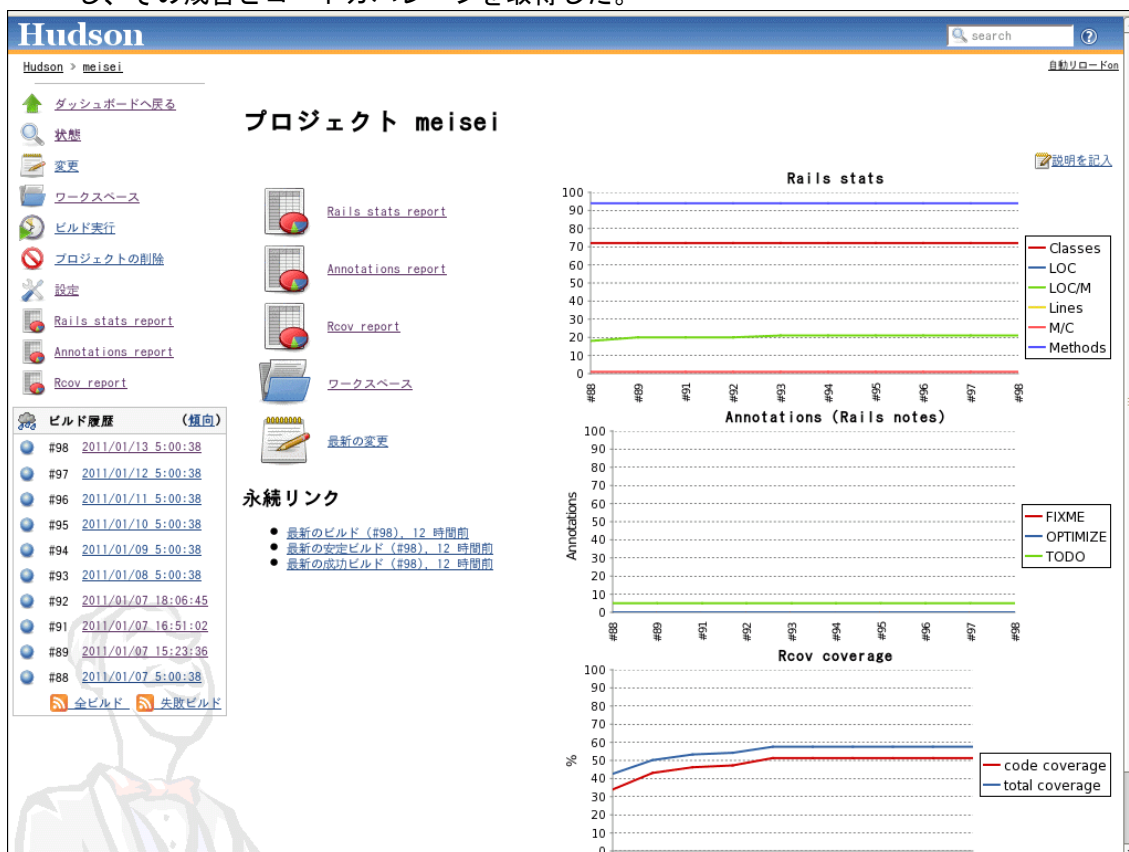


図 4-6 Hudson CI

(k) ユーザの参画

ユーザ側にも Web 会議システムを導入し、各拠点にしながらコミュニケーションを取れるようにした。また、分散開発環境にアクセスできる環境を用意し、システムを動かして確認できるようにしている。また、分散環境内のシステムを動かしながら、Web 会議で実装機能の説明や、スプリントレビューを実施した。

### (3) 開発管理におけるプラクティス

プロダクトバックログ、スプリントバックログ、およびバーンダウンチャートで開発管理を行った。また、日次のスクラムミーティングを実施し、作業内容の確認を行った。

#### (a) スクラムミーティング

分散開発環境（成果物確認に実装環境を用いた/コミュニケーションツールとして Web 会議システムを用いた）を利用して日次にて実施し、前日の作業内容、課題、当日の作業予定等を確認した。併せて、必要に応じて実装方法などに関する開発ディスカッションを実施し、分散開発環境上では説明しづらい箇所や仕様の検討を行った。

#### (b) プロダクトバックログ

システムを完成させるために必要な機能の一覧を登録し、管理した。プロダクトバックログには、ユーザからの“要望”と、実現する機能を書いた“ストーリー”を登録した。その上で、ユーザと協議して各ストーリーには実装優先度を設定し、実装のステータスを管理した。

#### (c) スプリントバックログ

ストーリー（機能）を実現するための作業一覧をスプリント計画で作成し、管理した。スプリントバックログは、スプリント開始前に開発チーム内で協議の上、プロダクトバックログから選択した。選択したストーリーを実現するための作業を“タスク”として登録し、各タスクの残作業時間（見積もり工数）を設定している。スプリントバックログには、作業のほかに、バグも登録した。

#### (d) バーンダウンチャート

スプリント毎の残作業時間グラフにより、プロジェクトの進捗を管理した。バーンダウンチャートは、日々更新され、その時点での残作業時間をグラフ化する。

#### (e) スプリントレビュー

スプリントの最後に、開発したシステムをユーザに見せ、開発した機能のレビューを実施した。レビューは、分散開発環境を使用して、画面共有にて実施した。

## 4-3 実証した開発手法、プラクティスの有効性と課題

### 4-3-1 システム規模

本事業で開発したシステムのステップ数を表 4-4 に示す。各コンポーネントの割合を図 4-7 に示す。もっとも大きいのは、Java と HTML である。本システムでは、PDF の帳票出力に Java のライブラリを使用しており、出力処理に必要な部分を Java で開発している。Java で開発した部分は、出力対象のデータを帳票テンプレートに埋め込む処理のみにも関わらず、ステップ数が多い結果となった。

表 4-4 ステップ数

#	言語	コンポーネント	ステップ数	言語別小計
1	Ruby	コントローラ	967	2,659
2		モデル	255	
3		ヘルパ	80	
4		プラグイン	191	
5		テストコード	543	
6		ビュー	623	
7	HTML	ビュー	2,335	2,335
8	JavaScript		169	169
9	CSS		303	303

10	Java	ライブラリ	2,349	2,349
	合計		7,805	7,805

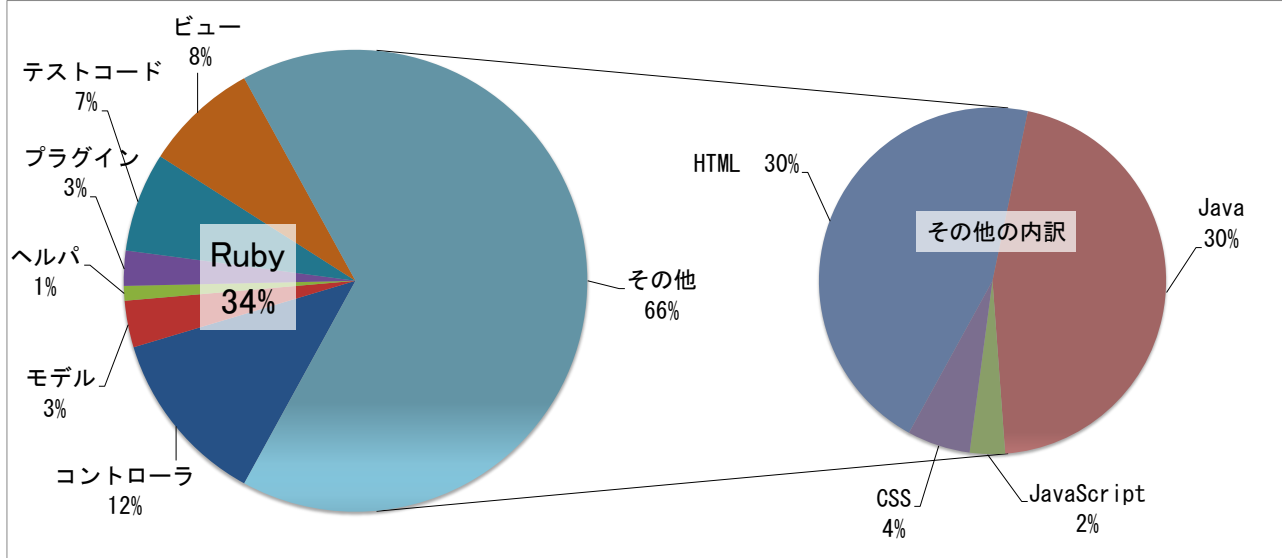


図 4-7 ステップ数の割合

テーブル数を表 4-5 に示す。

表 4-5 テーブル数

#	テーブル数
1	20

システムの画面数を表 4-6 に示す。

表 4-6 画面数

#	Web画面	帳票	計
1	23	6	29

#### 4-3-2 生産性

開発に要した工数を表 4-7 に示す。スプリント2において、タスクあたりの工数が多いのは、以下が原因である。

- ・ 帳票出力の実装に関する技術的課題解決に時間を要した
- ・ テストコードを用意するテスト駆動開発 (TDD) を試行による工数増加

表 4-7 開発工数

#	項目	スプリント1	スプリント2	スプリント3	計
1	ストーリー [件]	5	4	7	16
2	タスク数 [件]	35	22	49	106
3	工数 [時間]	103	180	160	443
4	タスクあたり工数 [時間/件]	2.9	8.2	3.3	4.2

生産性を数式 4-1、数式 4-2、数式 4-3、数式 4-4 に示す。全体としては、約 2 日で 1 画面を開発できる結果となった。

#### 数式 4-1 単位時間あたりのステップ数

$$\frac{7,805[\text{ステップ}]}{443[\text{時間}]} = 17.6[\text{ステップ/時間}]$$

#### 数式 4-2 1画面あたりのステップ数(すべてのコード)

$$\frac{7,805[\text{ステップ}]}{29[\text{画面}]} = 269.1[\text{ステップ/画面}]$$

#### 数式 4-3 画面あたりのステップ数(Rubyのみ)

$$\frac{2,659[\text{ステップ}]}{29[\text{画面}]} = 91.7[\text{ステップ/画面}]$$

#### 数式 4-4 1画面あたりの工数

$$\frac{443[\text{時間}]}{29[\text{画面}]} = 15.3[\text{時間/画面}]$$

#### 4-3-3 ソースコード

Ruby のソースコードの複雑さを測定するツール flog を使用して測定した結果を表 4-8 に示す。flog の結果は、“flog” という数値で出力され、値が大きいほど対象のコードは複雑である。複雑さの指標はないが、コントローラが最も複雑であり、リファクタリングの必要性があると考えられる。

表 4-8 ソースコードの複雑さ

#	項目	複雑さ [flog]	メソッドあたりの平均 [flog/メソッド]
1	コントローラ	1134.2	13.8
2	モデル	251.1	7.8
3	ヘルパ	64.8	10.8

安全にリファクタリングを行うには、テストコードが必要である。スプリント2において、テストコードを使用する TDD を試行した。しかし、Ruby on Rails のテストコードを記述する経験がなかったため、工数が増加する結果となった。機能の実現を優先するため、TDD は中止し、コードカバレッジは、47.25% に止まった。

また、Ruby 用のコードインスペクションツールを使用したコーディング規約の適用を実施したが、ツールの活用ができず、コーディング規約違反の修正ができていない。開発者が使用するツールが複数あり、trac、subversion 以外のツールに手が回らなかった点と、コードインスペクションがソースコードコミット後に実施されることからコーディング規約に強制力がなかった点が原因と考える。

#### 4-3-4 計画要求数/受入要求数/実現要求数

計画、受入、実現の各要求数を表 4-9 に示す。スプリント1、スプリント2において、受け入れた要求をスプリント内ですべて実現できず、次スプリントへの持ち越しが発生した。

主な原因は以下である。

- ・ 技術的課題の解決に時間を要した
- ・ 一ヶ月という開発サイクルを習得するまでに時間を要した
- ・ 分散開発環境への適応

スプリント1、スプリント2と連続して、受入要求をすべて実現できなかったことにより、スプリント2のリリース時、ユーザに納得して頂くのに時間を要する結果となった。

表 4-9 計画要求数/受入要求数/実現要求数

#	分類	スプリント1		スプリント2		スプリント3	
		ストーリー [件]	タスク [件]	ストーリー [件]	タスク [件]	ストーリー [件]	タスク [件]
1	計画	7	48	7	32	7	49
2	受入	7	51	7	37	7	49
3	実現	5	35	4	22	7	49

#### 4-3-5 割込(変更)の受入要求数/実現要求数

ユーザに対して、スプリント実施中でも分散開発環境を使用して、実際に動作させながら機能の説明を実施した。そこで得られたフィードバックをシステムに反映させながら開発を行っている。発生した変更要求の受入数と実現数を表 4-10 に示す。発生した変更要求の内、スプリント内で開発予定の機能で、かつ軽微な要求はタスクとしてスプリントバックログに追加し、スプリント内での実装を行った。画面追加などのストーリーレベルはプロダクトバックログへ追加した。

スプリント3で実装した和暦対応に 18[時間]を要したが、それ以外の割込要求に対しては、短時間で実現できる結果となった。

表 4-10 割込(変更)の受入要求数/実現要求数

#	項目	スプリント1	スプリント2	スプリント3 ( )は和暦対応 を除いた値	計
1	発生数 [件]	4	9	0	13
2	受入変更要求 [件]	3	5	5(4)	13
3	実現要求数 [件]	3	5	5(4)	13
4	工数 [時間]	4.0	4.0	32.0(14.0)	40.0
5	要求あたり平均工数 [時間/件]	1.3	0.8	6.4(2.3)	3.1

#### 4-3-6 有効性と課題

##### (1) 有効性

- (a) Ruby と Ruby on Rails を使用することで記述量が削減され、短期間に機能を作りこむことが可能
- (b) 和暦対応には時間を要したが、他の変更要求に対しては、平均 3.1 [時間/件] で要求を実装しており、機能の変更を短期間で対応できる

##### (2) 課題

- (a) 人的スキルの育成。開発メンバすべてが習得するの必要はないと考えるが、メンバの中に以下のスキルを持った人員が必要
  - (ア) Ruby の実践的な開発経験
  - (イ) Ruby 以外の JavaScript や Ajax、Database など Web システムに関連するスキルが必要
  - (ウ) 各種ツールを使いこなすスキルが必要
  - (エ) 一ヶ月という開発サイクルの習得
 開発者にとって、初めての手法、初めてのツール使用であったため、習得に時間を要した。また開発者が1名であったため、広範囲のスキルを必要とされ、スキル習得に時間を要した。こういったことが受入要求をすべて実現できなかった主な要因と考える。小規模開発では、少人数で開発を行うことになり、開発者は上記に挙げたスキルを習得しておく必要がある。
- (b) テストファースト開発のトレーニングおよび品質管理の確立  
安全なリファクタリングを行うには、テストコードが必要となる。しかし、テストコードを書くには Ruby on Rails アプリケーション開発の経験が必要であり、初心者には敷居が高い。テストコードを記述するトレーニングが必要と考える。また、開発者自身が品質全体を管理するのは実装にも影響し、負担が大きい。テストコードの妥当性やユーザ視点の考え方など、品質面を担うメンバが必要ではないかと考える。
- (c) コーディング規約の遵守  
ソースコミット前にチェックを行うなどの改善が必要

#### 4-4 実施体制の有効性と課題

##### 4-4-1 会議の実施結果

本事業の実施体制は、2-3 **実証体制（ユーザ含む）** に記載の通りで、3 機関により構成されている。また 4-2 (1) 分散開発環境に記載の通りで、地理的に離れた複数拠点をセキュアなネットワークで結んだ仮想的な共有環境を使用してプロジェクトの運営を行っている。このような環境で実装チーム/スクラムマスター/ユーザの 3 機関間で下記の通りで会議を実施した。尚、下記実績には 1 機関の内部会議はカウントしていない。

表 4-11 会議の実施結果

#	日付	会議名 ※定例進捗：実装チームと スクラムマスターとの 定例会議 (技術指導やふり返しを含む)	参加機関				会議方法 (対面・TV 会議)	
			ユ ー ザ	実装チーム		スクラムマ スター		
				東京	島根	東京		島根
1	2010/09/09	キックオフ	●	●	益田	益田	対面と TV 会議	
2	2010/09/10	実装チームコンサル			●	出雲	対面	
3	2010/09/17	定例進捗		●	●	●	TV 会議	

4	2010/09/24	定例進捗		●	●	●		TV 会議
5	2010/10/05	定例進捗		●	●	●		TV 会議
6	2010/10/05	スプリント 1 要件打合せ	●		益田			対面
7	2010/10/07	定例進捗		●	●	●		TV 会議
8	2010/10/21	定例進捗		●	●	●		TV 会議
9	2010/10/22	定例進捗		●	●	●		TV 会議
10	2010/10/28	定例進捗		●	●	●		TV 会議
		スプリント 1 リリースと スプリント 2 要件打合せ	●	●	益田	●		対面と TV 会議
11	2010/10/29	定例進捗		●	●	●		TV 会議
12	2010/11/04	定例進捗		●	●	●		TV 会議
13	2010/11/29	定例進捗		●	●	●		TV 会議
		スプリント 2 リリースと スプリント 3 要件打合せ	●	●	●	●		TV 会議
14	2010/12/01	定例進捗		●	●	●		TV 会議
15	2011/01/06	スプリント 3 リリースと スプリント 4 要件打合せ	●		益田			対面
16	2011/01/12	定例進捗		●	●	●		TV 会議
17	2011/01/31	スプリント 4 リリース	●		益田			対面

ユーザ参加の会議は、1 回 TV 会議のみ(ユーザ側に実装チームメンバやスクラムマスタメンバが出むかない)で実施した。

#### 4-4-2 コストについて

本事業の実施体制上、全プロジェクトメンバが 1 箇所に集うと一人当たり交通費だけで下記の通りコストが発生する。

表 4-12 本プロジェクトの交通費コスト

#	事例	交通手段 (往復交通費)	対象人数
1	ユーザ (益田明誠高校) で会議する場合	東京駅⇄益田 (¥73,680-)	2 名
		出雲市⇄益田 ( ¥4,420-)	2 名
2	実装チーム(出雲市)で会議する場合	東京駅⇄出雲市 (¥65,640-)	2 名
		益田⇄出雲市 ( ¥4,420-)	2 名
3	プロジェクトリーダー (東京) で会議する場合	益田⇄東京駅 (¥73,680-)	2 名
		出雲市⇄東京駅 (¥65,640-)	2 名

概算でも全プロジェクトメンバが1箇所に集うと交通費だけで¥140,120~¥278,640のコストが発生する。表 4-11 で示した会議のうち、実装チームとスクラムマスターとの会議を 12 回実施している。この会議を対面で行った場合、最も低コストの往復交通費(2 名移動 : ¥131,280-)で合計¥1,575,360-の費用が必要となる。この会議を今回全て TV 会議で実施した為、交通費に関しては一切費用が発生していない。TV 会議機材費用と通信費用は、今回既設の資産を流用した為、算出はしていないが、同じく本事業で算出していない移動時間に関する費用を考慮すると、本事業で利用した分散開発環境により、プロジェクト運営に掛かるコストを十分削減できると考える。

#### 4-4-3 有効性と課題

##### (1) 有効性

- (a) 地理的に離れた複数拠点でも、本事業の分散開発環境を利用することでプロジェクト運営は十分可能
- (b) プロジェクトリーダーと実装者が、拠点が離れていても、本事業の分散開発環境を利用して実施した本事業の開発手法のプラクティクスで、プロジェクト運営は十分可能
- (c) 本事業の分散開発環境を使用することで、会議実施で発生する交通費コストの大幅な削減を実現
- (d) 実装チーム内の会議については、TV 会議の利用経験値が高く、意思疎通も問題なくできた。よって TV 会議の利用経験値が高いメンバが参加する会議は、TV 会議のみの実施も可能

##### (2) 課題

- (a) ユーザ参加の会議で 1 回だけ実施したユーザ側に実装チームメンバやスクラムマスタメンバが出むかない会議で、ユーザと実装チームメンバとの意思疎通がうまく出来ない事象が発生した。よって、より円滑な会議を実施するために、TV 会議の利用経験値を上げることが必要
- (b) 会議の内容により、会議方法（対面か TV 会議）を検討することが必要

#### 4-5 ユーザ満足度に及ぼす有効性と課題

##### 4-5-1 開発したシステムに対する評価

実装チームが、評価シートをユーザへ配布し、開発したシステムについて評価を頂いた。次の通りの評価結果となった。

表 4-13 開発したシステムに対する評価

#	大項目	中項目	評価指標 1：全く満たしていない 2：一部満たしている 3：満たしている 4：満たした上に工夫がみられる。 5：満たした上にきわめて優れた工夫がみられる
1	ユーザビリティ	データの入力のし易さ	4
		情報のアクセス性	4
		情報のみつけ易さ	4
		情報の把握のし易さ	4
		情報のわかり易さ	3
		エラーの対処	3
2	レスポンス	反応のよさ	3
		動作のわかり易さ	4
3	機能		4
4	信頼性		5

開発したシステムについては、概ね満足頂いている。

##### 4-5-2 スケジュールと納期に対する評価

実装チームが、評価シートをユーザへ配布し、スケジュールと納期について評価を頂いた。次の通りの評価結果となった。

表 4-14 スケジュールと納期に対する評価

#	項目	対象スプリント	評価指標 1：全く満たしていない 2：一部満たしている 3：満たしている 4：満たした上に工夫がみられる。 5：満たした上にきわめて優れた工夫がみられる

1	納期準拠	スプリント1	3
		スプリント2	3
		スプリント3	4
		最終スプリント	4
2	スケジュール遅延時の対応	スプリント2	3
3	進捗報告	スプリント1	4
		スプリント2	4
		スプリント3	4
		最終スプリント	4

スケジュールと納期については、概ね満足頂いている。スケジュールが遅延したスプリント2の対応については、遅延した原因の説明とリカバリーについてユーザーに丁寧に説明したことを評価頂き、満足頂いている。

### 4-5-3 ドキュメントに対する評価

実装チームが、評価シートをユーザへ配布し、実装チームが作成しユーザへ配布した設計書並びに仕様書、その他の説明書、会議アジェンダ資料を対象として、評価を頂いた。次の通りの評価結果となった。

表 4-15 ドキュメントに対する評価

#	項目	媒体種類	評価指標 1：全く満たしていない 2：一部満たしている 3：満たしている 4：満たした上に工夫がみられる。 5：満たした上にきわめて優れた工夫がみられる
1	文章の質	ドキュメント	4
		電子メール	4
2	文章のわかり易さ	ドキュメント	4
		電子メール	4
3	レイアウト	ドキュメント	3
4	図表	ドキュメント	4
5	索引	ドキュメント	3
6	用語解説	ドキュメント	3

ドキュメントについては、概ね満足頂いている。後述コメントに「専門用語が多い」というコメントがあるが、理解できない用語に関しては、都度ユーザから実装チームに確認して明確になったとのコメントを頂いた為、ご理解頂いていると判断している。

### 4-5-4 イテレーション（スプリント）に対する評価

実装チームが、評価シートをユーザへ配布し、イテレーション（スプリント）について評価を頂いた。次の通りの評価結果となった。

表 4-16 イテレーション（スプリント）に対する評価

#	項目	対象スプリント	評価指標 1：全く満たしていない 2：一部満たしている 3：満たしている 4：満たした上に工夫がみられる。 5：満たした上にきわめて優れた工夫がみられる
1	実施のタイミング	スプリント1	3
		スプリント2	3
		スプリント3	3
		最終スプリント	3
2	実施方法	スプリント1	2
		スプリント2	2
		スプリント3	4
		最終スプリント	4

イテレーション（スプリント）については、実施のタイミング（各リリースとリリース間の日数）については概ね満足頂いている。実施方法については、スプリント1とスプリント2については、一部満足頂いていない結果となった。スプリント1は、開発チーム側の不慣れ（経験不足）が原因、2スプリントは、スケジュール遅れが原因である。また両スプリント共に受入要件全てを実現できなかったことも一部満足頂けなかった原因である。

#### 4-5-5 コミュニケーションに対する評価

評価シートをユーザへ配布し、コミュニケーションについて評価を頂いた。次の通りの評価結果となった。

表 4-17 コミュニケーションに対する評価

#	項目	対象スプリント	評価指標 1：全く満たしていない 2：一部満たしている 3：満たしている 4：満たした上に工夫がみられる。 5：満たした上にきわめて優れた工夫がみられる
1	実施方法	電話	3
		電子メール	3
		TV 会議	2
		対面	4
2	意思疎通		3
3	仕様の確認	スプリント1	3
		スプリント2	3
		スプリント3	4
		最終スプリント	4
4	対応の柔軟性	スプリント1	3
		スプリント2	3
		スプリント3	4
		最終スプリント	4

コミュニケーションについては、TV 会議以外については概ね満足頂いている。TV 会議については、使用方法の複雑性や実施方法、視覚範囲の仕様のな要因により、ユーザが不快を感じたことが原因と考える。他の項目については満足頂いているが、スプリント1とスプリント2については、イテレーション（スプリント）の評価結果と関係して他スプリントと比較して、満足度が低くなったと考える。

#### 4-5-6 元々の課題に対する評価

評価シートをユーザへ配布し、元々の課題について評価を頂いた。次の通りの評価結果となった。

表 4-18 元々の課題に対する評価

#	大項目	中項目・ 対象スプリント	評価指標 1：全く満たしていない 2：一部満たしている 3：満たしている 4：満たした上に工夫がみられる。 5：満たした上にきわめて優れた工夫がみられる
1	広域募集により、 手動では管理しきれない	生徒数の増大に耐えられるシステム	4
		仕事量の増大に耐えられるシステム	5
		管理し易い	4
2	必要な機能が未知である	スプリントにより必要な機能が抽出できた	4
3	早期に事業を軌道	すぐにも新事業が実施可	4

	にのせたい	能なシステム	
4	事業拡大に伴って 判明する必要機能を 順次追加したい	スプリント1	3
		スプリント2	3
		スプリント3	4
		最終スプリント	4

元々の課題については、各項目概ね満足頂いている。特に新規事業の特性の1つである広域募集による業務増大に対して、システムが耐えられるという評価が“5”であったことは、開発したシステム自体の評価は勿論のこと、本プロジェクトで選択した開発手法で抽出した機能の積み重ねが実現した結果だと考える。

#### 4-5-7 コストパフォーマンスに対する評価

評価シートをユーザへ配布し、コストパフォーマンスについて評価を頂いた。次の通りの評価結果となった。

表 4-19 コストパフォーマンスに対する評価

#	項目	評価指標
		1：全く満たしていない 2：一部満たしている 3：満たしている 4：満たした上に工夫がみられる。 5：満たした上にきわめて優れた工夫がみられる
1	開発コスト	3
2	参画コスト	3

コストパフォーマンスについては、満足頂いている結果ではあるが、評価して頂いたユーザ側リーダー並びに主担当者がコストオーナーではなかったことを考慮する必要がある。

#### 4-5-8 有効性と課題

##### (1) 有効性

- (a) 本事業のユーザから頂いた評価から、「必要な機能が未知である」「早期に事業を軌道にのせたい」「事業拡大によって判明する必要機能を順次追加したい」ユーザに対して、本事業で適用した言語(Ruby)と開発手法(インクリメンタル開発)は、ユーザ満足度を得られる開発言語並びに開発手法である
- (b) 本事業のユーザ(高等学校)の都合で1週間~2週間毎のイテレーションが困難であった。よってイテレーション計画を1ヶ月としたことと、打合せ時間を放課後(16時以降)に設定したことでユーザが受入し易いタイミングを設定したことで、ユーザ満足度を得られた
- (c) 分散開発でも「ユーザとの距離があることによる苦情はなかった」ことから、ユーザ満足度への影響はなく、スムーズなプロジェクト運営は可能

##### (2) 課題

- (a) コストパフォーマンスについて、ユーザ担当者がコストオーナーでない場合に満足度をあげる取り組みが必要と考える。受入要件確定時の提示情報にコスト情報を盛り込む必要があると考える
- (b) 進捗報告については、開発環境で利用する開発者向けのツールでは伝わりにくいと考える。今回は、ユーザ向けには進捗報告資料(計画要求/受入要求/実現要求それぞれの件数と内容を記載した資料)を改めて作成したために、満足度は良かった。よって今後も同様の方法がよいと考える
- (c) スケジュールの遅延時並びに計画要件と実現要件で差異が発生した場合(計画要件を全て満たさなかった場合)の対応方法で工夫が必要。特に計画要件と実現要件に差異が発生した場合の理由や原因の説明には、ユーザが分かり易い情報の提示が必要である。(b)に述べた内容とは正反対となるが、日々のリアルタイムの進捗状況をユーザ側に提示していれば、ここで述べる課題が解決する可能性がある

## 4-6 品質・信頼性及び開発管理に及ぼす有効性と課題

### 4-6-1 品質と信頼性

品質管理は、定期的なウォークスルーレビューと機能実装完了後のテストツールによる画面操作テストを行った。表 4-20 に作成テストシナリオ数と工数を示す。記録済みテストシナリオを再利用することにより、新規に作成するシナリオ数は減る傾向にあるが、画面修正により、テストシナリオの修正が必要になったことでテストに要する工数が増加している。

表 4-20 作成テストシナリオ数と工数

#	項目	スプリント1	スプリント2	スプリント3	計
1	作成テストシナリオ [件]	44	28	17	89
2	工数 [時間]	20.0	40.0	43.5	102.5

各スプリントで発生したバグ数と修正時間を表 4-21 に、単位規模あたりのバグ数を数式 4-5 に示す。バグは短時間で修正できており、スプリントを重ねても修正時間の増加はない。

表 4-21 バグ発生件数と修正時間

#	項目	スプリント1	スプリント2	スプリント3	計
1	バグ [件]	3	3	9	15
2	工数 [時間]	5.0	4.0	10.0	19.0
3	バグあたり [時間/件]	1.7	1.3	1.1	1.3

数式 4-5 単位規模あたりのバグ数

$$\frac{15[\text{件}]}{7.8[\text{Kステップ}]} = 1.92[\text{件/Kステップ}]$$

バグを修正するために修正規模を表 4-22 に示す。スプリント2の修正規模が他と比較して大きいのは、画面の修正を行ったためである。画面の修正には HTML を変更する必要があるため、その結果、修正箇所が多くなっている。

表 4-22 バグの修正規模

#	項目	スプリント1	スプリント2	スプリント3	計
1	バグ [件]	3	3	9	15
2	修正ステップ数 [ステップ]	12	42	35	89
3	修正規模 [ステップ/件]	4.0	14.0	3.9	5.9

発生したバグの種類を図 4-8 に示す。最も多いのは「仕様の抜け」である。システムの仕様は、ユーザとのコミュニケーションで決定しているが、細かな仕様までは確認しきれなかった。コミュニケーションのみでの伝達には限界があると考えられる。

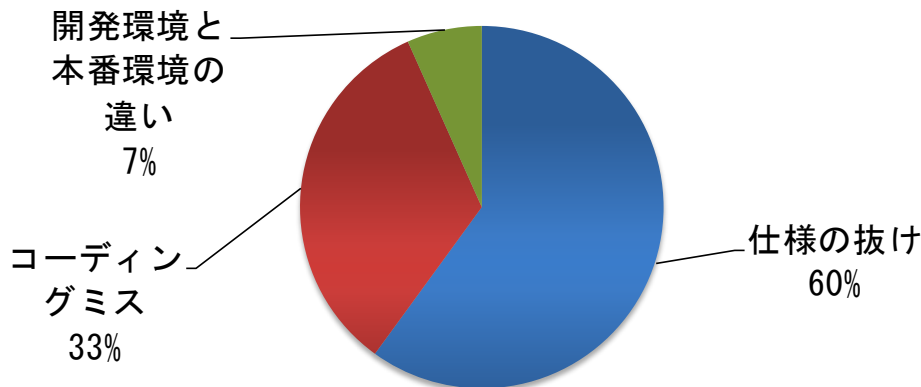


図 4-8 バグの種類

### 4-6-2 分散開発環境を使用することの影響

本事業では、実装はプロビズモ出雲本社において1名で実施した。ウォークスルーレビューのレビューアは、HISOL 側においてリモートでレビューを実施している。Ruby による記述量が少なさと小規模開発であ

った点で、ソースコードのチェックが可能であった。今回に関しては、分散開発環境に起因する品質劣化はなく、品質への影響はないと考える。

ウォークスルーレビューによる指摘事項は、trac にタスクとして登録し、実施状況を確認できるようにした。また、trac と構成管理ツールである subversion を連携させ、ソースの修正箇所を確認できるようにした。trac や subversion といったツールを活用することにより、コードの修正をトレースできる。

また、ユーザが使用するシステムも分散環境内に配置したことにより、修正内容の反映を迅速に行うことが可能であった。

#### 4-6-3 品質・信頼性の有効性と課題

##### (3) 有効性

- (d) 平均 1.3 [時間/件] でバグの修正ができており、短時間でバグの修正が可能
- (e) イテレーションが進み、システム規模が増加しても修正時間の増加がない
- (f) 分散開発でも品質への影響はなく、コード修正内容のトレースとシステムへの修正反映が迅速に可能

##### (4) 課題

- (a) テスト工数の軽減。画面の変更が頻繁に行われるため、画面操作テストスクリプトの修正工数が増加する

#### 4-6-4 開発管理

開発管理は、プロダクトバックログ、スプリントバックログ、スプリントバーンダウンチャートで行った。分散開発のため、紙ではなく、trac を用いて行った（図 4-9）。trac には、Scrum 用プラグインである agile を適用し、スプリントバーンダウンチャートの自動生成を実現している。

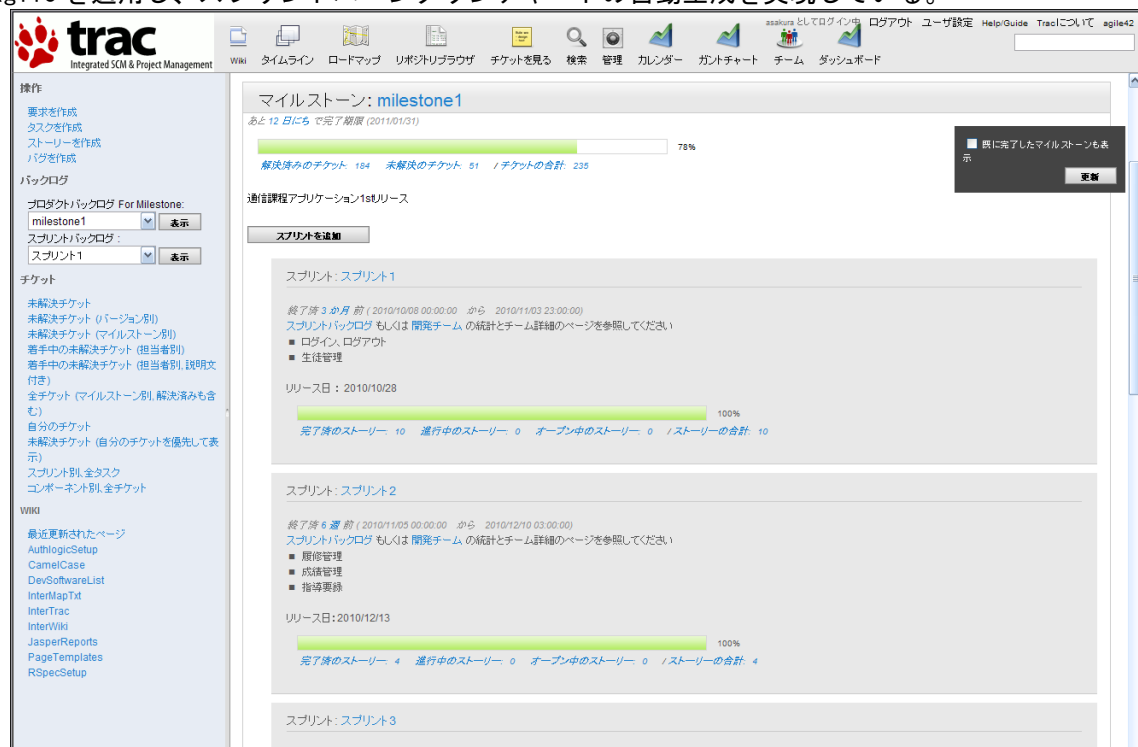


図 4-9 agile を適用した trac

##### (1) スクラムミーティング

プロビズモ出雲本社とプロビズモ東京支社間を TV 会議で接続し、開発者、ユーザ窓口、リーダ間で日次スクラムミーティングを実施し、各自の前日の作業内容、課題、当日の作業予定等を確認した。また必要時、隔日で開発ディスカッションを実施した。ミーティングでは各ツールに記録した進捗情報を活用した。

##### (2) プロダクトバックログ

システムに必要な機能をストーリーとしてプロダクトバックログに登録し管理を行った（図 4-10）。このプロダクトバックログを元にユーザとのミーティングを実施し、機能の優先順位を決定している。優

先順位は、「重要度」として、高い順に“必須”、“あると良い”、“魅力的”の3段階で設定した。

The screenshot shows the Trac interface for a product backlog. The main area displays a table of tasks for 'milestone1'. The tasks are sorted by priority, with '魅力的' (Charismatic) at the top, followed by 'あると良い' (Nice to have), and '必須' (Must have) at the bottom. The table includes columns for ID, 概要 (Summary), 重要度 (Priority), コンポーネント (Component), スプリント (Sprint), ストーリーポイント (Story Points), and 担当者 (Assignee).

ID	概要	重要度	コンポーネント	スプリント	ストーリーポイント	担当者
#234	設定情報の入力名を可読とする					n.a.
#344	ユーザおよび生徒の所属を管理できる	あると良い	90 システム管理	スプリント4	3	h-nishimura
#90	学籍マスタメンテナンスの閲覧	あると良い	90 システム管理	スプリント4	2	h-nishimura
#91	教科マスタメンテナンスの閲覧	あると良い	90 システム管理	スプリント4	2	h-nishimura
#92	科目マスタメンテナンスの閲覧	あると良い	90 システム管理	スプリント4	2	h-nishimura
#93	教材マスタメンテナンスの閲覧	あると良い	90 システム管理	スプリント4	2	h-nishimura
#95	ユーザマスタメンテナンスの閲覧	あると良い	90 システム管理	スプリント4	3	h-nishimura
#79	費用請求の閲覧<基本機能>	あると良い	70 費用請求	スプリント3	8	dev01
#99	履修計画情報 科目のコード入力の実現	あると良い	40 履修計画情報	n.a.	2	h-nishimura
#89	調査書の閲覧	あると良い	80 調査書	n.a.	5	h-nishimura
#167	住所マスタメンテナンスの閲覧	魅力的	90 システム管理	n.a.	3	n.a.
#37	生徒一覧の閲覧	必須	10 生徒一覧	スプリント1	3	h-nishimura
#44	生徒基本情報の閲覧	必須	20 生徒基本情報	スプリント1	3	h-nishimura
#96	生徒基本情報-年次項目ドロップダウン化	必須	20 生徒基本情報	スプリント1	1	h-nishimura
#98	生徒基本情報-携帯電話番号項目の追加	必須	20 生徒基本情報	スプリント1	1	h-nishimura
#97	生徒基本情報-年齢項目の追加	必須	20 生徒基本情報	スプリント1	1	h-nishimura
#49	保護者情報の閲覧	必須	30 保護者情報	スプリント1	3	h-nishimura
#26	ロガー機能の閲覧	必須	90 システム管理	スプリント1	2	h-nishimura
#7	ログアウト機能の閲覧	必須	90 システム管理	スプリント1	1	h-nishimura
#103	住所検索の閲覧	必須	95 共通機能	スプリント1	1	h-nishimura

図 4-10 プロダクトバックログ

各ストーリーには、以下の情報を設定し、機能の実現状況を管理した（図 4-11）。

- ・ 実現する機能の「概要」と「説明」
- ・ 機能の「重要度」、「完了」、「実施中」、「未着手」などの「ステータス」
- ・ 機能を実装する「担当者」
- ・ 実装を行う「スプリント」

The screenshot shows the details of a story titled '保護者情報の開発' (Development of Parent Information). The story is assigned to 'okada' and is in the 'closed' state. It is linked to '30 保護者情報' in the '90 システム管理' component and 'スプリント1'. The priority is set to '必須' (Must have). The story points are 3. The assignee is 'h-nishimura'. The description states: '生徒に対応する保護者情報の登録、編集ができる。' (Registration and editing of parent information corresponding to students). A list of related tasks is provided, including '表示機能の実装', '登録機能の実装', '更新機能の実装', '削除機能の実装', '保護者情報モデリング', 'テストスク립トの作成', and 'テストシナリオの作成'.

図 4-11 機能を表わすストーリー

スプリント実施中で発生した割込み要求もプロダクトバックログに追加し、次のスプリント計画時に優先順位の再設定を行った。これにより、ユーザが重要と考える機能から優先的に実装することができ、ユーザにとって価値のある機能を作りこむことができたと考える。

変更要求が発生したため、当初予定の機能より増加したが、Rubyの生産性によって“必須”とされる機能は、開発期間内で実装することができた。ただし、“あると良い”、“魅力的”といった機能に未実装のものがある。予め用意した予備のスプリントで対応したが、増加する要望への対応が課題である。

### (3) スプリントバックログ

スプリント開始前にユーザを含めた開発チーム内で一ヶ月間の開発範囲を決定し、スプリントバックログを作成した。スプリントバックログには、プロダクトバックログからストーリーを選択し、そのストーリーを実現するためのタスクを登録した(図 4-12)。タスクは、1日以内で完了できる粒度に分割し、タスクごとに残作業時間を見積もった。ストーリーと同じく、“完了”、“実施中”、“未着手”などのステータスを設定し、タスクの消化状況を可視化した(図 4-13)。

ID	概要	ストーリーポイント	スプリント	状況	実工数(時間)	コンポーネント	残存時間	担当者	開始日	終了日
#242	費用請求の開発(教材費)	2	スプリント4	closed		70 費用請求		h-nishimura	2011/1/11	2011/1/11
#292	テストケースの修正		スプリント4	closed	1	70 費用請求	0.0h	h-nishimura	2011/1/11	2011/1/11
#291	テストシナリオの修正		スプリント4	closed	1	70 費用請求	0.0h	h-nishimura	2011/1/11	2011/1/11
#293	教材費出力が広実装		スプリント4	closed	4	70 費用請求	0.0h	h-nishimura	2011/1/11	2011/1/11
#294	テストとテストスクリプト修正		スプリント4	closed	4	70 費用請求	0.0h	h-nishimura	2011/1/11	2011/1/12
#351	システム全体の機能改善	3	スプリント4	assigned		95 共通機能		h-nishimura		
#241	年を選択しないでサブミットする と'undefined method 'name' for nil:NilClass"		スプリント4	new	n.a.	95 共通機能	1.0h	h-nishimura		
#342	observe_fieldのonと .frequencyのオプションを削除		スプリント4	closed	0.5	95 共通機能	0.0h	h-nishimura	n.a.	n.a.
#357	生徒基本情報を削除した時の dependentが古いものに 戻っている		スプリント4	closed	0.5	20 生徒基本情報	0.0h	h-nishimura	n.a.	n.a.
#358	履修計画PDF出力でMS 明		スプリント4	closed	n.a.	95 共通機能	0.0h	h-nishimura	n.a.	n.a.

図 4-12 スプリントバックログ

**更新機能の実装** (登録: 3か月前, 最終更新: 3か月前)

報告者: okada | 担当者: h-nishimura  
 コンポーネント: 20 生徒基本情報 | キーワード: n.a.  
 スプリント: スプリント1 | リソース: n.a.  
 残存時間: 0.0h | 開始日: 2010/10/14  
 終了日: 2010/10/14 | ガントに含める: 1  
 進捗率(%): 100 | 依存関係(チケット): n.a.  
 実工数(時間): 8

**説明**

参照

参照元  
 - ストーリー (#44): 生徒基本情報の開発

**添付ファイル**

**チケットの履歴**

更新者: okada (3か月前)

- 担当者 somebody から h-nishimura に変更されました
- ガントに含める が設定されました。

更新者: okada (3か月前)

- 概要 が 生徒基本情報変更の更新 から 生徒基本情報変更機能の更新 に変更されました。

図 4-13 タスク

スプリント2までは、複数のタスクに着手するなどタスクの完了が遅れ、消化状況を確認できないことがあった。タスク間の依存がないようにタスクを分割するといったタスク分割の工夫や開発手法の習得により、改善することができた。

(4) スプリントバーンダウンチャート

スプリントにおける開発の進捗管理は、tracが表示するバーンダウンチャート(残作業時間グラフ)で管理した(図4-14)。バーンダウンチャートは、tracによって自動的に更新され、特定時点の残作業時間と今後の消化傾向が表示される。

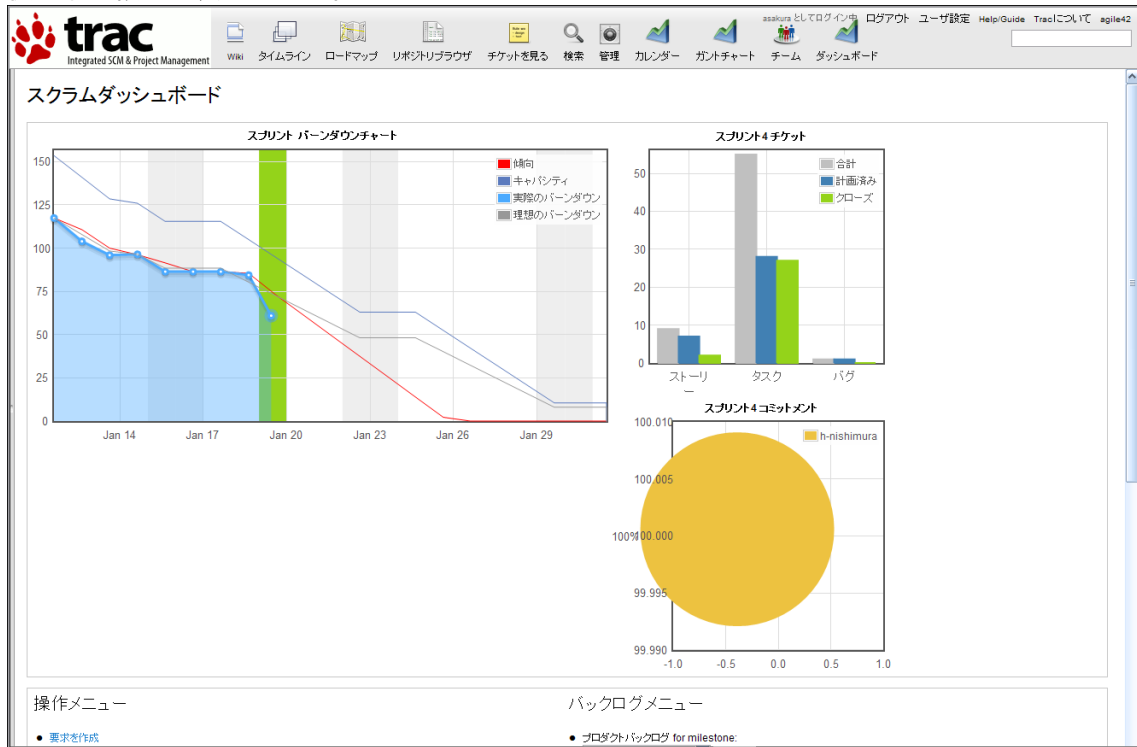


図 4-14 バーンダウンチャート

以下、各スプリントのバーンダウンチャートを示す。

(a) スプリント1

tracの不調により、タスクのステータス変更ができず、実際の開発状況との差異が発生し、バーンダウンチャートでの進捗管理ができていない。そのため、trac外でのタスク消化状況の確認と日々のスクラムミーティングにて進捗管理を代用した。

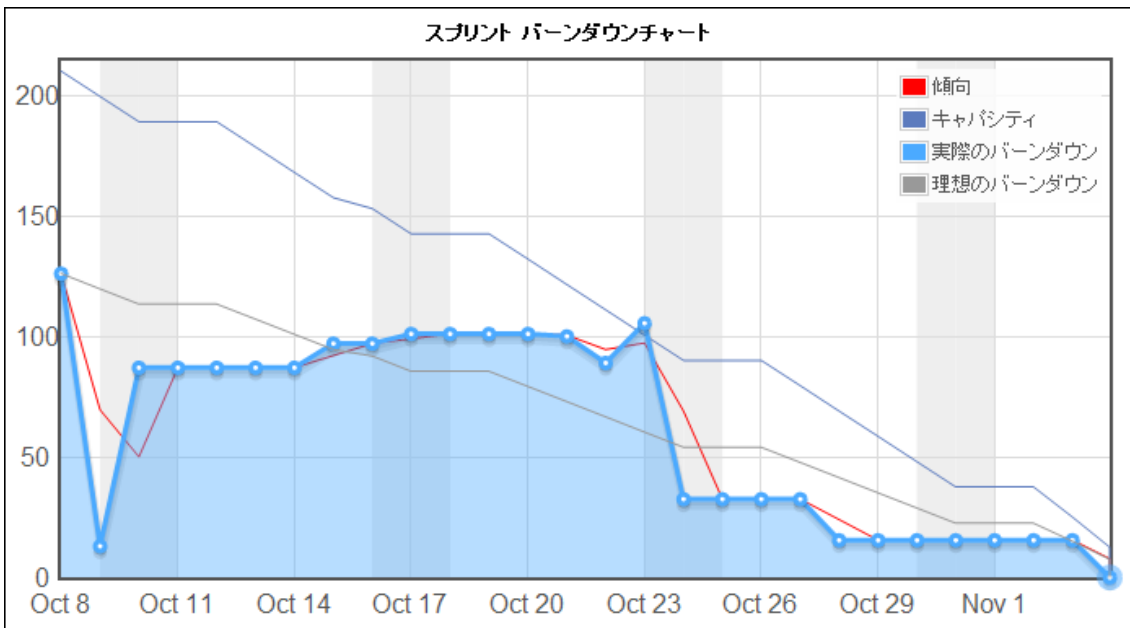


図 4-15 スプリント1のバーンダウンチャート

(b) スプリント2

スプリント開始時点で、各タスクの残作業時間が設定されていなかったため、理想線である“理想のバーンダウン”が利用できない状況となった。また、複数のタスクが“着手中”のままとなり、バーンダウンチャートからは進捗が確認できない状況となった。

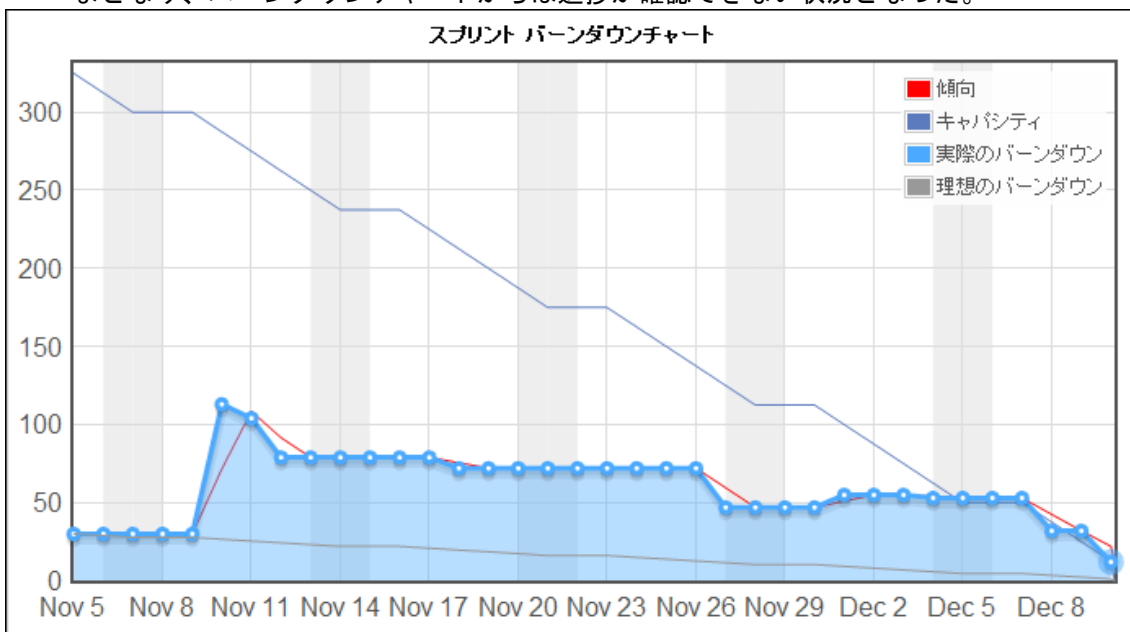


図 4-16 スプリント2のバーンダウンチャート

(c) スプリント3

開発環境が安定し、開発チームの習熟度も上がり、逐次タスクのステータスが更新できた点と、過去のスプリントの失敗を踏まえたタスク分割の工夫を行ったことにより、バーンダウンチャートによる進捗管理ができるようになった。

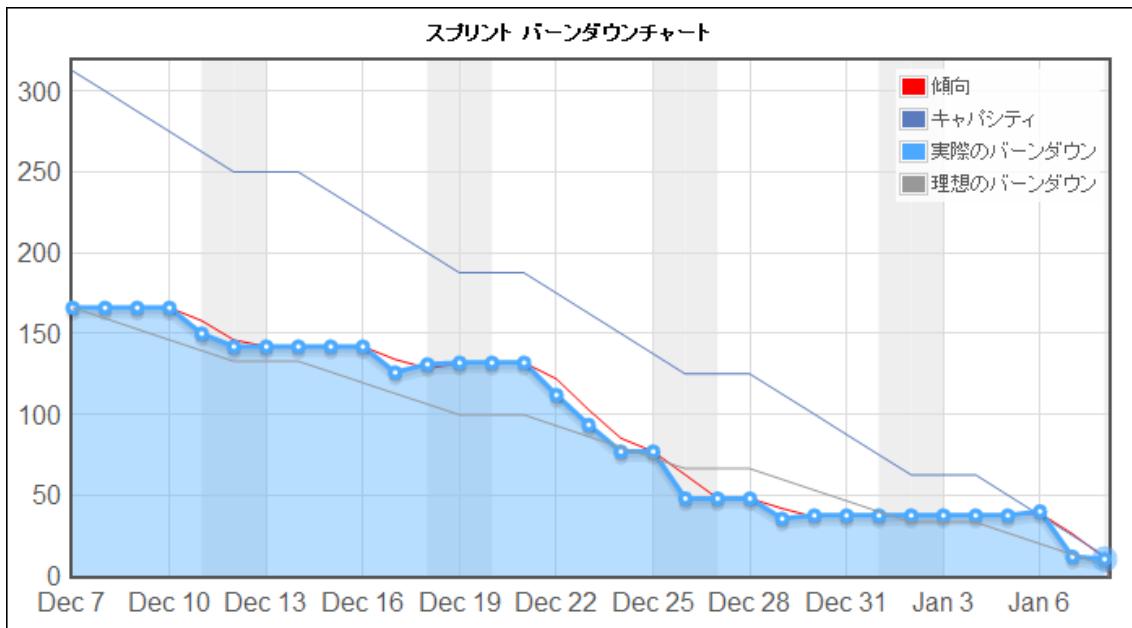


図 4-17 スプリント 3 のバーンダウンチャート

#### 4-6-5 開発管理の有効性と課題

##### (1) 有効性

- (a) 優先順位を明確にすることにより、ユーザにとって価値のある機能から作りこむことが可能
- (b) ツールを活用することにより、開発管理に必要なバーンダウンチャートの自動化が可能
- (c) trac 上のバックログの消化状況およびバーンダウンチャートによって、進捗管理が可能
- (d) ツールを使うことにより、分散開発でもメンバが開発状況を確認することが可能

##### (2) 課題

- (a) 増加する要望への対応
- (b) 開発環境の安定性の確保
- (c) ツールを使いこなすスキルの習得  
タスクの分割方法や、日々の開発サイクルに適應できないとバーンダウンチャートが意味のないものになってしまう。このように、正確な情報が登録されないと、遠隔地にいるメンバの状況を把握することができない。
- (d) ユーザ視点での「見える化」  
遠隔地にいるユーザに対し、trac を用いて開発の進捗を報告したが、trac は開発者向けのものであり、ユーザにとっては分かり難いものであった。実装する機能、もしくは実装される機能がユーザにとってどのような価値があるのかなど、ユーザの視点で管理できるしくみが必要である。

#### 4-7 開発側参加者の意見

開発側参加者のコメントは、次の通りである。

表 4-23 開発側コメント

#	コメント
1	<p>【Ruby 技術の習得について】</p> <p>今回、短期間で Ruby 技術の習得できたのは、経験豊富なスクラムマスターのコーチングが大きかった。具体的なサンプルコードでの説明など理解しやすい工夫が多くあった。</p> <p>スクラムマスターが先頭に立ってより最適な実装方法などを調査・検討することで、または障害などの問題解決を行うことで、開発者の Ruby の理解、本実装時の時間短縮やバグ発生率の低下など円滑な開発につながったと考える。</p> <p>ユーザ様への打合せ（要件ヒアリング）では、Ruby 技術の習得の未熟さから具体的には Ruby ではどう実装するかわからないまま提案を行い、後日技術的裏付けをとる場面が多かった。</p> <p>想定したよりも開発内容と期間がシビアだった為、全体的な理解よりも実装を優先した為、今回使用してない部分の習得はできず、リファクタリングも十分には行えなかった。</p>

2	<p>【インクリメンタル開発手法の習得について】</p> <p>本事業のようなケース（当初完成予想及び要件決定がほとんどされてなく、ユーザが業務をかかえ要件定義に時間がさけないようなケース）ではとても有効な開発手法と感じた。</p> <p>スプリント途中又はスプリント完了後に、不確定だった要件がより鮮明に見える形になる為、眠っているユーザ要件を掘り起こし易い。その反面、細かな部分の変更が多発して本来の目的である機能実装に影響が出てしまう可能性もありタスク管理（タスクの優先度付けなど）をしっかりとる必要があった。</p> <p>また、どうしても要件のトレードオフが発生してしまう為、ユーザ側の満足度（あれもこれもできるだけ実装してほしい）を得にくい部分もあると感じる。</p>
3	<p>【ユーザとのやりとりなどで感じた思い等】</p> <p>打合せ決定して実装しても、変更があり実装方法まで大きく影響する変更に関しては手戻りが発生しているようにも感じた。</p> <p>従来の開発手法より、ユーザが開発に踏み込める分、より技術的な裏付けや作業工数の算出根拠が明確に必要なになる。</p> <p>どうしてもユーザからの要望が膨らんでしまいがちになる危険性がある。（当初に開発完了イメージが未確定の為）</p> <p>開発手法上の専門用語が特殊で（例：スプリント、インクリメンタル開発等）、ユーザに意味が伝わりにくいのでより理解しやすい言葉へ置き換え説明するように心がける必要があった。</p> <p>プロジェクト開始当初、プロジェクトを円滑に進めるためには、ユーザとの関係をよりよくする必要がありと考え、ついユーザの要求を取り入れようとし過ぎたことで、ユーザに過剰な期待感や満足感を抱かせてしまい、結果一部ユーザ満足度を下げってしまうことにつながってしまった。ユーザとの関係をよりよくする場合でも、ビジネスとしてしっかりとした一線を意識して活動にあたる必要があると考える。</p>

#### 4-8 ユーザ側参加者の意見

実装チームが、ユーザ満足度評価シートをユーザへ配布した際に頂いたユーザ側参加者のコメントは、次の通りである。1に記載が好意的評価、2に記載が好意的でない評価コメントである。

表 4-24 ユーザコメント

#	コメント
1	<p>対応が誠実だった。</p> <p>手間のかかることでも、誠実に提案等して頂きより良いシステム構築につながった。</p> <p>不確定なことが多い状況での構築着手にも関わらず、この方式ゆえ完成できたのではないかと思う。</p>
2	<p>専門用語が多く、イメージしにくかった。</p> <p>TV 会議はあまり意味がなかったようにも思う。</p> <p>成果物が曖昧なまま、開発が進んだ。</p> <p>帳票デザイン力がなかった。デザイン力もプロフェッショナルだと思っていた。</p> <p>システム構築が初めてだったので、アジャイルや Ruby 言語の優位性はよくわからない。</p> <p>途中でのテストラン、回数が少なかった。</p> <p>スクラムマスターの動きがわかりにくかった。</p>

またスプリント3終了時に本事業についての感想をお聞きした際コメントは、次の通りである。

表 4-25 スプリント3終了時のコメント

#	コメント
1	100%の満足はしていない。またアジャイルの優位性がはっきりとしないが、ユーザのニーズを掘り起こすには、今回の手法はいい方法であると考える。更に小さい機能から積み上げる今回の方法は、この事業で開発したシステムやビジネスの特性に適していたほうでないか。
2	本事業スタート時、正直漠然とプロジェクトを開始したので、期待度はそれほどなかった。それにもかかわらず開発したシステムや成果がここまでの形になったことは、今回適用した手法が間違いではなかったように思う。
3	実現要件の検討時に、各機能の価格がわかると良かった。要件が受けいれられない場合に、価格や工数があふれるのか、開発側の力量不足なのかがわからなかった。
4	本スプリントで、新規事業で利用するシステムの頭蓋骨や背骨、ほとんどの骨は完成した。今後はどのような服を着せるかが課題となる。ユーザ側だけではなく、開発側にも一緒にどのような服を着せるかを考えてほしい。また着せる服は、都度変わる可能性がある。
5	今回の事業で、新規事業におけるシステムの ABS やエアバック装置はないが、アクセルやブレーキはできた。今後も ABS やエアバック装置は求めないが、アクセルやブレーキのアップグレードを進めたい。

## 5 総括

### 5-1 Ruby の特徴を活かす開発手法の今後のビジネス利用の可能性と活用分野

Ruby を使用したインクリメンタル開発によって、短期間で機能の実装と変更要求への対応が可能である。要件があいまいな小規模システムをユーザのフィードバックを取り入れながら少しずつ開発する場合に有効ではないかと考える。当社は、今回の開発手法を利用することにより、以下に挙げるような Web システムの開発に活用できると考える。

- ・ 新規事業向け IT システムや情報系システムなどリリース時期が重要で早く実現したいシステム  
すべての要件が決定されていなくとも、1 回のスプリント分を確定すれば開発を進めることができ、プロジェクト途中でも運用を開始することができるため、ビジネス開始時期が最重要とされるシステムに向いていると考える。
- ・ イン트라ネットシステムや CRM、SFA など費用対効果が求められるシステム  
実際にシステムを動かして、効果を確認しながら少しずつ開発できる。システムとして必要だが、無駄な投資を避けたいシステム開発に向いていると考える。
- ・ B2C などユーザの意見をシステムに取り込みたいシステム  
変更要求への柔軟性と短期間での機能実装が可能であるため、システムを利用する側の意見を取り込みやすいと考える。

また、分散開発環境を利用することで、遠隔地のユーザでもプロジェクトに参加することが可能である。

### 5-2 Ruby の特徴を活かす開発手法の今後のビジネス利用での課題と克服策

- (1) 大規模システム開発への対応  
開発者が多い大規模システム開発にどう適用するかが課題である。チーム構成や品質管理など大規模システム開発時の手法を検討する必要がある。
- (2) インクリメンタル開発を始めとするアジャイル開発経験者の不足  
アジャイル開発においては、プロセスを実践するだけでは上手くいかず、アジャイル開発の経験が必要である。実際に経験することが重要であるが、本事業の結果を参考にして頂くことで、これからアジャイル開発を行う企業の手助けとなると考える。
- (3) 変更要求の増加  
請負契約において、要件変更増加に対するコスト増の対応が課題である。要件変更増加による修正コ

ストを抑えるプロセスが必要と考える。

- (4) インクリメンタル開発を始めとするアジャイル開発手法のユーザ側の理解不足  
担当者は勿論のこと、コストオーナーも含めユーザ側が、アジャイル開発についての知識を有することが必要である。プロジェクト開始の勉強会等でのフォローも有効的ではあるが、円滑にアジャイル開発手法を用いてプロジェクトを推進するには、一定のレベルに達することができる外部研修受講をタスクとして取り入れることも必要と考える。

### 5-3 今回開発システムの今後のビジネス展開の可能性課題と克服策

#### (1) 標準業務フローの策定

本事業で開発した通信過程向け教務システムが対象とする業務フローは、使用する学校によって異なる場合が多い。今後のビジネス展開を考えると標準業務フローを策定し、これを推奨する必要があると考える。

#### (2) 複数校共同利用

本事業で開発した通信過程向け教務システムが対象とする標準業務フローが策定されると、この標準業務フローを受け入れ可能な複数学校が共同で利用することが求められる。複数校共同利用を目的としたセンターサーバ的な対応が必要になる。このような形で使用されるシステムとして必要な業務仕様や機能仕様の検討、開発が必要である。

#### (3) 法令改定の対応方法

本事業で開発した通信過程向け教務システムが出力する帳票の一部は、法令改定に伴う変更時に、専門的な技術が必要である。今後は、法令改定に伴う変更時に、学校教職員が簡単に変更できる仕組みが必要である。

### 5-4 ビジネス利用での見積・契約での提言

#### 見積もり

- (1) 見積もり方法は、リファクタリングにより、常にソースコードは見直されるためステップカウント法での見積もりは合わない。機能規模を基本にした手法で見積もるのが良いと考える。機能規模を数値化する手法としては、ファンクションポイント法、ストーリーポイントなどがある。
- (2) 請負型のシステム開発において、イテレーション(スプリント)毎に見積もりを行うことは難しい。開発の前にシステムの要件を整理する計画フェーズを設け、初期見積もりを行う。
- (3) 変更要求や新たな要望が発生した場合は、再見積もりを行う。再見積もりのタイミングとしてはイテレーションの区切りなどが挙げられる。

#### 契約

- (1) 契約は、初期見積もり額での契約を基本とし、初期見積もり額に、バッファとして変更分を上乗せした額を予算とする。
- (2) 発注者と開発側は、修正コストがバッファを超えないよう最大限の努力を行う。
- (3) 変更要求が発生した場合は、イテレーション毎に変更分を再見積もりし、必要に応じて再契約を行う。
- (4) 発注者側で要件、要望を挙げるメンバにコストの権限を与えることが望ましい。

### 5-5 人材育成に関する提言

- (1) 開発者にもコミュニケーション能力が求められる。
- (2) インクリメンタル開発を始めとするアジャイル開発手法は、実際に経験しないと身に付かない。
- (3) Rubyを使ったシステム開発では、書籍や教育ではすべてをカバーすることはできないため、インターネット上の情報に頼らざるを得ない。情報の真偽の判断能力、数多くあるライブラリの選別能力や有益性などを見極める能力が必要とされる。
- (4) プロジェクト参加者には、使用する開発環境を有効活用できる能力が必要とされる。
- (5) アジャイル開発手法を適用する場合は、ユーザ側コストオーナーや担当者もアジャイル開発手法を理解しておく必要がある。開発者側からの説明のみではなく、研修受講や試行などが必要である。