Ruby on Rails によるチーム開発の授業実践

高橋圭一[†1]

概要：2016 年度に本学科の 1 年次向けのプログラミング言語を Java から Ruby に切り替えた．そのため，3 年次の Web アプリケーション開発の講義および演習で使用する開発フレームワークを Java EE から Ruby on Rails に変更した．本稿では，2018 年度の後期に実施した Ruby on Rails によるチーム開発の授業内容について報告する．受講生は前期に Ruby on Rails の基礎を学習済みである．そのため後期には，Session やモデル間の関連付けなど Ruby on Rails の発展的な機能を学び，チーム開発で必要となる Git，Bitbucket，Heroku などのツールを学習したあと，2 名ずつの 8 チームで 8 週間かけて開発を進めた．結果としては，各チームが開発したソースコードは平均で約 1400 行であり，J2EE を使用したときと同規模となった．一方，画面数および画面遷移数は 2018 年度の方が前年より上回っており，Ruby on Rails の様々な支援機構により，より実用的なアプリケーションが開発できたことがわかった．

キーワード：Ruby，プログラミング演習，開発フレームワーク

The class practice of team software development with Ruby on Rails

KEIICHI TAKAHASHI[†1]

Abstract: In 2016, we changed the introductory programming language in our department from Java to Ruby. Because of this change, we also changed the web application development framework from Java EE to Ruby on Rails, which has been used in lectures and exercises of the 3rd year. In this paper, we report the class practice of team software development with Ruby on Rails in the second semester in 2018. The students learn the basics of Ruby on Rails in the first semester of 2018. Therefore, in the second semester, they learn advanced features of Ruby on Rails and tools such as Git, Bitbucket, and Heroku, and then start team development. As a result, the source code developed by each team is about 1,400 lines, on average, which is the same size as when using J2EE. On the other hand, the number of webpages and webpage transitions was higher in 2018 than in the previous year, and it was found that various support mechanisms of Ruby on Rails could develop more practical applications.

Keywords: Ruby, Programming Exercise, Development Frameworks, Team Software Development

1. 背景

本学科は 2004 年度に設立した定員 70 名の学科であり，ACM /IEEE-CS が策定したコンピューティング・カリキュラム（CC-2001）をもとに設計したカリキュラムにより，ネットワークやソフトウェアの企画・設計・開発・保守などに関わる技術者の輩出を目指してきた．2016 年度に，時代の変化に合わせて CG や Web などのメディア制作やデータサイエンスのコースを加えてカリキュラムを改定した．

カリキュラム改定前には，導入から応用まで一貫してプログラミング言語として Java を採用してきたが，カリキュラム改定後には，プログラミングを必ずしも課さないコースもあるため，1 年前期のみプログラミング科目を必修とし，Java より”おまじない”が少ないため簡潔に記述できるプログラミング言語として Ruby を選択した．また，学部の所在地である福岡県が Ruby の振興に力を入れており，在学中や卒業後に学生が地元のコミュニティに参加するきっかけになると考えたことも理由の 1 つである．なお，所属コースによっては選択科目も含まれるが，プログラミング言語としては，1 年後期に Processing（≒Java），2 年次に C#と C++を学習する科目があり，データサイエンスコースの科目では R や Python を学習する機会がある．2 年後期にはデータベース科目で SQL を学習する．

本稿で対象とする 3 年次の Web アプリケーション開発の科目を図 1 に示す．前期には，Web アプリケーション開発の基本的な技術を学ぶ科目として，講義と演習を 1 コマずつ連続した時間割で実施する．後期には，前期の学習内容をもとに，発展的な内容を学習したあと，2〜3 名でチームを編成して独自の Web アプリケーションを開発する．カリキュラム改定前では，統合開発環境 Eclipse[1]および Java の Web アプリケーションフレームワークである Java EE（Java Platform Enterprise Edition）を採用していたが，Ruby への変更に合わせて，開発環境を Cloud9 IDE に[2]，Web アプリケーションフレームワークを Ruby on Rails（以降 Rails と略す）に変更した．カリキュラム改定後の 3 年次前期の授業については報告済みであるため[3]，本稿では，2018 年度の後期に実施した「ソフトウェア開発演習」（図 1 着色部）の授業実施内容について報告する．

†1 近畿大学産業理工学部情報学科
Kindai University

©2019 Information Processing Society of Japan

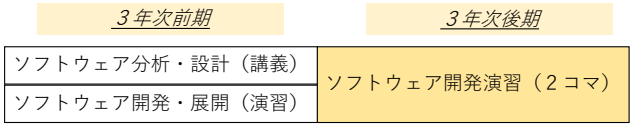

図 1　3 年次の Web アプリケーション開発の科目

2. 授業計画

2.1 学習内容

カリキュラム改定後のソフトウェア開発演習の授業計画を図 2 に示す．本科目は 2 コマ 4 単位で，エンジニア系コースの必修科目である．指導者は教員が著者 1 名，TA1 名である．

授業の流れ：第 1 回目に前期の学習内容を復習し，第 2 回から第 6 回目までは，前期では扱わなかった Rails に関する発展的な内容について学習する．特にチーム開発を進める上で必要となるバージョン管理 Git[4]，リモートリポジトリ Bitbucket[5]の使い方や，実用的なアプリケーションを開発する上で必要となる認証機能やモデルの関連付けなどを学ぶ．なお，PaaS（Platform as a Service）の 1 つである Heroku[6]についてはチーム開発では必ずしも必要ないが，開発した Web アプリケーションを就職活動でのポートフォリオとして利用することがあるため，使用方法を解説して活用させる．チーム開発は第 7 回から第 14 回まで行う．第 7 回にユーザストーリとモデルとワイヤーフレームの簡単な演習を行ったあと，各チームで開発する Web アプリケーションの企画を話し合いまとめる．なお，開発する Web アプリケーションの内容や機能については特に制約は設けない代わりに，企画時に指導者が積極的に関わり助言することで目標とする機能や品質を担保する．

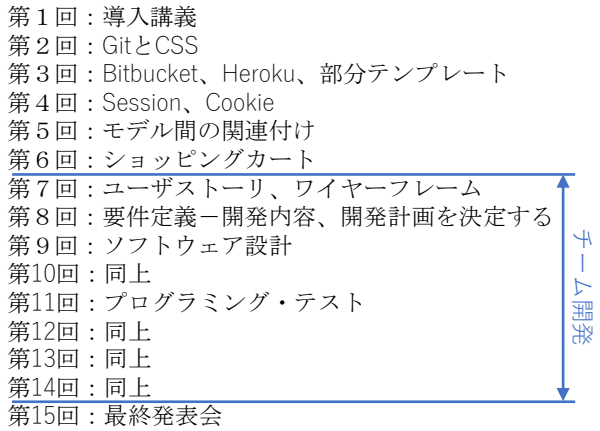

図 2　ソフトウェア開発演習の授業計画

成績評価基準: レポート課題 (30%), 開発システム (40%), チーム貢献度（20%），プレゼンテーション（10%）で評価する．レポート課題は，第 1 回から第 6 回までプログラミング課題があるため，個人ごとにレポートを評価する．第 7 回以降はチームごとの作業報告を各自のレポート課題として評価する．開発システムは設計書や最終発表から機能やフールプルーフや使いやすさの配慮の有無を読み取り，各チームの実力を勘案して評価する．チーム貢献度は，Git のコミットログからメンバーごとのコミット行数 *LOC* を取得し，(1)式で求めるチーム貢献率が 1 を大きく下回った場合，評価点 20 点にチーム貢献率を乗じて評価点とする．貢献度に大きな差がない場合は 20 点を与える．

$$\text{チーム貢献率}_c = \frac{n \cdot LOC_c}{\sum_i^n LOC_i} \qquad (1)$$

チーム編成：前期演習科目の成績順に 2 名ずつチームを作る．2018 年度は履修者が 17 名（男性 14 名，女性 3 名）であったため，成績上位者から 2 名ずつのチームが 7 チーム，成績下位者の 3 名のチームが 1 チームとなった．成績最下位者の 3 名のうち 2 名は前期の科目を受講していなかったが，受講を強く希望したため 3 名でチームを組ませた．

カリキュラム改定前の授業計画：比較のため，カリキュラム改定前の授業計画を図 3 に示す．授業の流れはほぼ同じである．第 1 回から第 6 回まで Java Servlet や Java Beans を活用した JSP Model2[7]など発展的な内容を学び，第 7 回以降に 2〜3 名でチーム開発する．評価方法もほぼ同様である．ただし，Git，Bitbucket，Heroku といったツールやサービスは利用せず，統合開発環境 Eclipse 上でプログラムコードを開発し，バージョン管理はせずチーム内でソースファイルを手動で共有して開発を進めた．

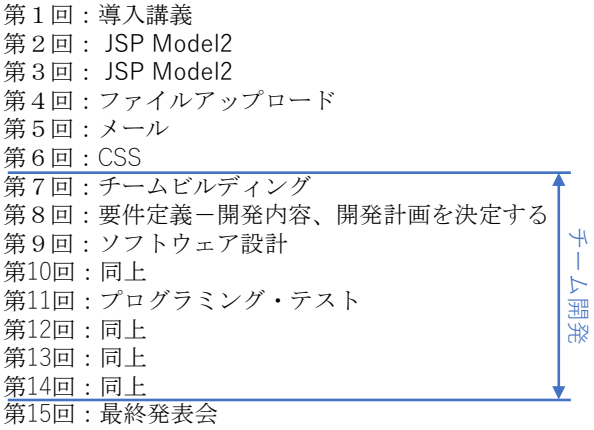

図 3 ソフトウェア開発演習の授業計画（カリキュラム改定前）

2.2 Rails の特徴

Rails の基本的な特徴について説明する．Rails は Ruby で書かれたオープンソースの Web アプリケーションフレームワークである。公開されてから 15 年近く経過しており，GitHub やクックパッドなど国内外に多くの利用者を抱えるサービスでも採用されている．Rails チュートリアル[8]などの学習用のオンライン・ドキュメントも充実しており，また，Ruby の開発者が日本人であるため，日本語で読めるリソースがインターネット上に豊富にあり国内の初学者が学びやすいことも特徴の 1 つである．

Rails は基本哲学として DRY（Don’t Repeat Yourself；同じ

©2019 Information Processing Society of Japan

ことを繰り返さない)と CoC(Convention over Configuration;設定より規約）を掲げており，Rails が採用している MVC（Model View Controller）アーキテクチャを理解し，rails コマンドなどの各種ツールを使いこなすことで品質の高いソフトウェアを効率的に開発できる．図 4 に Rails の基本的な構成を示す．Rails アプリケーションが Web ブラウザから HTTP リクエストを受信すると，ルーティング(routes.rb)に書かれたコントローラ（Ruby クラス）を呼び出し，対応するビュー（ERB ファイル[a]）やモデル（Ruby クラス）を呼び出す．これらのファイルは rails コマンドにより雛形が自動生成され，クラス定義など定型的なコードの記述を省略できる．これらファイルを Rails が定めた場所に配置することにより，Java EE のように設定用の XML ファイルを記述することなく呼び出すことができる．こうした様々な規約が Rails の DRY や CoC といった基本哲学の表れである．

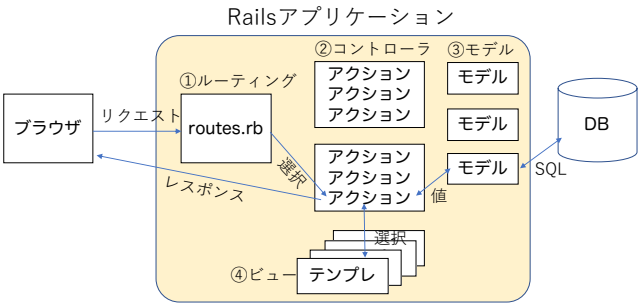

図 4　Rails の MVC アーキテクチャ

2.3 Rails を採用する上での懸念事項

Java EE から Rails に変更する上で以下のような 3 つの懸念事項があった．

（１）　Rails の仕組みを理解して開発を進められるのか

図 5 に本講義で扱う Java EE と Rails の主な構成要素を示す．Java EE でも，JSP や JDBC(Java Database Connectivity)などの要素をブラックボックスとして扱うが，各要素を組み合わせるプログラムコードは手動で記述する必要がある．一方，Rails では定型的な処理フローを手動で記述する手間を省くため，自動生成機能や支援機能があり（着色部），Java EE と比べるとブラックボックスの範囲が大きい．そのため，受講生が追加したプログラムコードとは無関係と思える箇所でのエラーメッセージが表示されることがあり，デバッグ作業を困難になる．受講生らがこれら Rails の仕組みを理解して開発を進められるか懸念された．

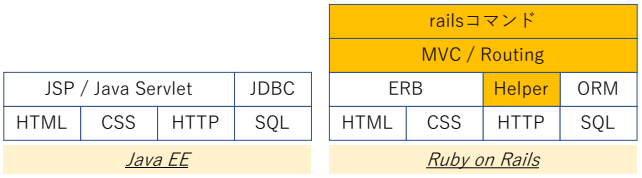

図 5　本講義で扱う Java EE と Rails の主な構成要素

（２）　Git や Bitbucket などのバージョン管理ツールを使いこなし，ブランチ開発ができるのか

近年のソフトウェア開発ではバージョン管理ツールの使用は標準的になっている．統合開発環境 Eclipse でもバージョン管理やチーム開発のための支援機能があるが，受講生の多くはチーム開発に慣れていないため，開発作業にできるだけ集中させた方がよいと判断して採用を見送ってきた．一方，Rails でのチーム開発では，こうしたツールの利用は必須とも言えるため，バージョン管理ツール Git およびチーム開発支援のためのリモートリポジトリ Bitbucket を採用することにしたが，チーム開発の作業を混乱させて開発作業を停滞させてしまう懸念があった．

（３）　Rails という特定のツールを学ぶことを Web アプリケーション開発を学ぶこととしてよいのか

ソフトウェア開発技術はまさに日進月歩である．特にインターネットやモバイルの周辺技術はフレームワークや支援ツールのみならず，プログラミング言語自体も流行り廃れが早い．Rails は誕生して 15 年近く経過する安定した技術の 1 つであるため，すぐに不要な技術と見なされることはないが，本講義で学習するのは，あくまで Rails を用いた Web アプリケーション開発である．カリキュラム改定前の Java EE では，プログラミング言語や開発環境が多少変わっても使える技術を扱ってきた．例えば，HTTP リクエストを受信して，その内容に応じて処理を分岐するアルゴリズムは素朴ではあるが基本的な処理の流れである．データベース処理のために SQL 文字列を記述して JDBC などのインタフェースでアクセスする方法も他のプログラム言語や開発環境に対応づけることが容易である．近い将来に受講生らが Rails 以外の方法で Web アプリケーションを構築するときに本科目で学習したことは役立つのだろうかという懸念があった．

2.4 環境構築

本学科の演習室には約 150 台のパソコン（Windows10）があり，ネットワークブート方式のシンクライアント・システムとして動作している．本方式では，アプリケーションなどはディスクイメージとして固定されており，アプリケーションの設定情報などのプロファイル情報は，ユーザ毎にネットワーク上のファイルサーバで管理される．

Linux や macOS と比べると Windows での Rails 開発環境の構築は難しいと言われており，実際にユーザ情報がファイルサーバー上で分散管理されるシンクライアント・システムでは以下のような試行を行ってみたがうまく動作させることができなかった．最終的にクラウドベースの統合開発環境を利用することとした．

- ローカルドライブにインストール
- ネットワークドライブにインストール

a Ruby コードを埋め込める HTML ファイル

©2019 Information Processing Society of Japan

- 仮想化ソフトウェアを利用してインストール

クラウドベースの統合開発環境は，Web ブラウザのみで利用できる開発環境でプログラムの編集・実行はもちろんターミナルが利用できるため，ローカルパソコンと同様の感覚で Linux コマンドを実行することができる．このクラウドベースの統合開発環境の 1 つである Cloud9 社による Cloud9 IDE は基本的に無料で利用できるが，アカウント登録時にクレジットカード情報を入力する必要があるため採用が困難であった．2016 年に同社が開始した Cloud9 for Education というサービスを利用することでこの問題を回避できるようになったため，本稿で対象とする講義・演習では本サービスを利用することとしたb.

3. 授業実施結果

3.1 1回の演習の流れ

コンピュータ演習室に集合して，チームごとに直接会話できるように横並びに着席して開発を進めた．2 コマ連続して開講する科目のため，授業 1 回あたりの作業時間は 3 時間である．作業の成果物や作業中に発生した問題点を発表資料にまとめ，作業終了時刻の 30 分前からチームごとに作業報告をさせた．作業報告の冒頭に進捗状況（図 6）を含めるように指示した．これは残りの作業項目と作業完了に要する時間をチーム内で共有するためである．その後，メンバーごとに演習当日の作業内容についてスクリーンショットを用いて（図 7），効率的かつ要点を押さえた報告をするように指導した．作業報告時に設計上考慮すべき問題点があれば理由を添えて指摘し，前向きな解決方法を助言するなど，作業報告が形骸化しないように努めた．

全体の作業内容

作業内容	担当	今日の作業(分)	進捗(%)	残り作業時間(時間)
基本設計	Y、M		100	0
状態遷移図	Y、M		100	0
ワイヤーフレーム	M		90	0.25
モデル設計	Y、M		100	0
ログイン、ログアウト機能	Y		100	0
ホーム、マイページ	Y	80	85->87	2.5
チャット機能	Y	80	85->87	2.5
取引確認のQRコード	M	90	70->90	1
サイトデザイン	Y、M	90	0	3
バリデーション、入力エラー処理	Y	20	0->30	2

図 6 作業項目一覧表（作業報告冒頭部）

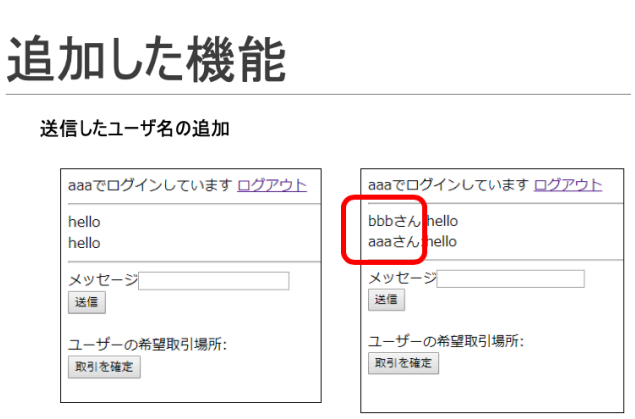

図 7 個人別の作業報告の具体例

3.2 最終成果物

科目の最終回にチーム毎に最終発表会を実施した．そのときに以下の 2 つのファイルを提出させた．

- 基本設計書（以下の項目を含む）
  - Bitbucket と Heroku の URL
  - 開発目的と対象ユーザ
  - システムの機能と特徴
  - ユーザ・ストーリー
  - モデル
  - ワイヤーフレーム
  - ページ遷移図
- 発表資料（PowerPoint ファイル）

基本設計書から抽出した各チームの成果物に関する情報を表 1 に示す．画面数と遷移数はページ遷移図から読み取った値である．LOC は各チームのリモートリポジトリ Bitbucket よりソースコードを入手し，モデル（/app/models），ビュー（/app/views），コントローラ（/app/controllers）のディレクトリ内にある Ruby と ERB ファイルの行数をカウントした値である．Rails のソースコードには自動生成されたファイルが数多く含まれるため，それらファイルをカウントから除外するためである．なお，前期成績が最下位であったチーム H はシステム自体を完成させて最終発表会にも参加したが，ソースコードの提出がなかったため空欄となっている．

表 1 の画面数と遷移数にばらつきが見られる．これらの数値は開発に対する自信に比例するとも考えられるが，受講生らに「設計したアプリケーションをすべて実装することは求めない」と伝えたので，その理由で差が生じたとは考えにくい．それより，各チームが企画したアプリケーションの性質による差異と考える．例えば，E と G はそれぞれ JavaScript で自作したドローツールとオンラインゲームを利用させる Web アプリケーションを企画した．そのため Ruby で実装する画面数は他と比べて少なくなったのではないかと考える．

b Cloud9 は 2019 年 12 月 31 日にサービス停止がアナウンスされたため，2019 年度の本科目では AWS Educate[9]というサービスを利用して受講生らにアカウント登録させ AWS Cloud9 を利用している．

©2019 Information Processing Society of Japan

また，表 1 の各チームの LOC にもばらつきが見られる．内訳は省略するが，メディア系コースの受講生がいるチームでは，設計したデザインを実現するため 1000 行以上の CSS の記述したため，この差が生じたようである．LOC の中央値は 1431 であるが，授業時間 3 時間に加えて授業時間外には平均で 3 時間作業したようであるため（表 5），作業時間は{3+3(時間)}×8(回)×2(人)で 96(人時)となる．生産性は 1431(LOC)÷96(人時)で約 15 (LOC/人時)となる．プロフェッショナルの案件と比較するのは難しいかもしれないが，ソフトウェアデータ白書[10]によると，新規開発の生産性の中央値は 15〜41(LOC/人時)である．開発経験の少ない受講生らではあるが，極端に低い値ではなかったと言える．

表 1　各チームの成果物に関する情報

チーム	システム名称	人数	画面数	遷移数	LOC
A	レシピ共有サイト	2	14	18	3601
B	スケジュール管理帳	2	15	25	1431
C	時間割アプリ	2	15	18	1586
D	学内商品交換システム	2	14	15	2004
E	PictShare	2	8	13	690
F	福岡探索マップ	2	8	16	1295
G	オンラインマッチングゲーム	2	6	10	558
H	フリーマーケット方式の物々交換サイト	3	8	15	-

3.3 各チームのコミット状況

各チームの活動状況を調べるため，リモートリポジトリである Bitbucket より取得した授業週ごとの各チームのコミット数を表 2 に示す．最初の 2 回は開発するシステムの企画がなかなか決まらずコーディングに入れないチームが多かったためコミット数は少なめである．中盤にはコードの修正や削除など動きが出始め，終盤の 3 回は各チームともコミット数が多くなった．全チームの総コミット数 326 をチーム数 7×実施週数 12 週で割ると，授業 1 回当たりのコミット数は 3.9 であった．したがって，平均すると各チームのメンバーは授業ごとに 1 コミットしたことになる．

表 2　授業実施週ごとの各チームのコミット数

授業回	A	B	C	D	E	F	G
7	0	2	11	0	0	0	0
8	0	0	6	3	0	0	0
9	7	3	3	0	0	9	0
10	12	2	4	2	0	3	0
11	5	6	4	2	0	0	0
12	0	6	3	1	0	2	0
13	20	3	17	4	1	8	0
14	2	4	7	10	0	10	0
15	33	12	13	36	15	32	3
合計	79	38	68	58	16	64	3

3.4 gem の利用状況

gem は正式には RubyGems[11]といい，Ruby 用のライブラリパッケージ管理システムであり，様々な有用なライブラリが登録されている．Ruby や Rails の開発者が開発・登録し，使い方などをインターネット上に様々な形で公開している．ログイン機能やファイルアップロード機能はスクラッチから Ruby で記述する方法を演習しており，その知識を利用して実装することを想定していたが，学生らがチーム開発の中で情報収集して devise や carrierwave をはじめ，bootstrap などのデザインテンプレートも積極的に採用していたことがわかった．

表 3　各チームが採用した gem ライブラリ

チーム	gemライブラリ			
	devise ログイン処理	carrierwave 画像アップロード	bootstrap CSSテンプレート	will_paginate ページネーション
A	✔	✔	✔	✔
B				
C			✔	
D				
E		✔		✔
F		✔	✔	✔
G	✔		✔	

4. 考察

4.1 懸念事項の検証

本節では，2.3 節に挙げた計画段階で想定していた懸念事項について，2018 年度の授業実施結果をもとに考察する．

（1） Rails の仕組みを理解して開発を進められるのか

Rails の仕組みを理解できなかったのであれば Web アプリケーションは完成しなかったはずである．チームによって規模や機能の差はあるものの，前期科目を履修しなかった最下位のチーム含め，Web アプリケーションを完成できたことから Rails の仕組みを理解できたと考える．ただ，どの程度の成果物を完成したのかわからないため，カリキュラム改定前の 2016・2017 年度の成果物と比較する．2016・2017 年度のチーム数は合わせて 12 であった．まず，コード行数の分布を比較する．カリキュラム改定前にはバージョン管理ツールを採用していなかったため，分析に利用したソースコードは 5 チーム分のみである．図 8 より，カリキュラム改定前後で第 1〜3 四分位数はほぼ同じ分布であった．つまり，Java EE で自作したコード量と Rails の支援機能を活用して自作したコード量が同じであったということになる．受講生のレベルは大きく変化していないはずであるから，この結果から少なくとも受講生らは Java EE と同じくらい Rails の基本機能を使いこなしていたと考えられる．

©2019 Information Processing Society of Japan

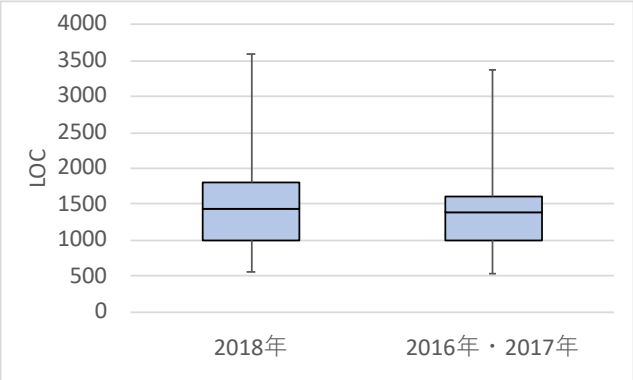

図 8　カリキュラム改定前後のコード行数の比較

画面数，遷移数の分布についても比較する（図 9，図 10）．画面数と遷移数については 2018 年度の方が多いことがわかる．つまり，カリキュラム改定前後で受講生らが記述したコード量は同規模であったが，2018 年度の方がより機能の多い Web アプリケーションを開発したことになる．

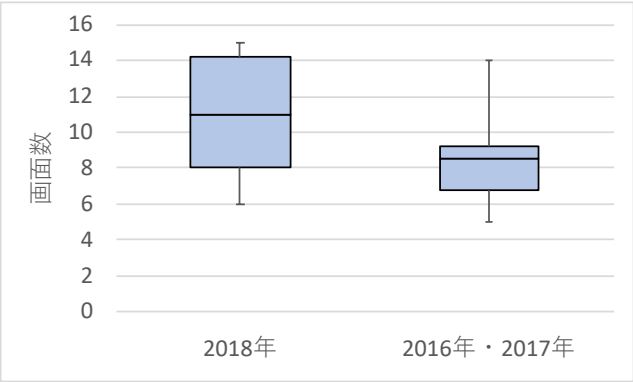

図 9　カリキュラム改定前後の画面数の比較

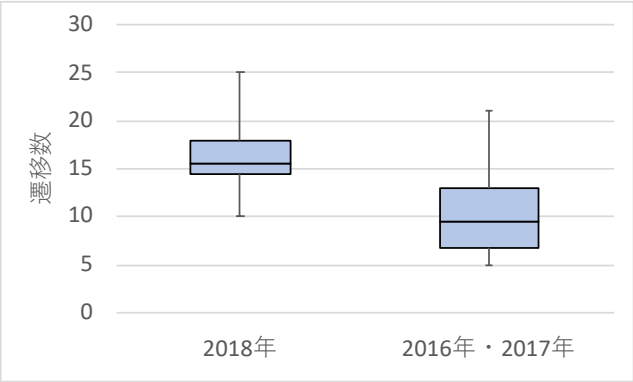

図 10　カリキュラム改定前後の遷移数の比較

（２）　Git や Bitbucket などのバージョン管理ツールを使いこなし，ブランチ開発ができるのか

3.3 節の表 2 より，チーム開発の序盤ではチームによってコミット数にばらつきがあったが，コミット総数を開発期間で平均すると各チームのメンバーが授業ごとに 1 コミットしたことがわかった．Git と Bitbucket については概ね活用できていたと言えるだろう．ただし，チーム E と G については終盤しか Git を利用しなかったようである．チームメンバーに確認したところ表 4 に示す回答があった．いずれも作業報告時点で気づいて助言あるいは指導できたケースであった．今後，リポジトリの利用状況を報告させるなど改善したい．

表 4　Git の利用頻度が少なかった理由

チーム	回答
E	Cloud9の共有ワークスペース機能を使って共同開発していたためGitは利用しなかった．評価にGitを用いると知って慌ててリポジトリを作成した．
G	ソースコードをBitbucketで管理していたが，終盤で動かなくなってリポジトリを再作成した．お互いにコミットできているかわからなくなり終盤でやっと理解できた．

また，講師として指導していて約半数くらいのチームが開発ブランチを master ブランチにマージできないトラブルを経験していた．また，Heroku にプッシュするときに gem ライブラリの不一致で失敗する事象があった．これらのトラブルについては指導時に問題解決を優先してしまい詳しい状況が残っていない．今後，使用するツールやサービスについてトラブルの状況と対応方法を記録して受講生に公開していきたい．

（３）　Rails という特定のツールを学ぶことを Web アプリケーション開発を学ぶこととしてよいのか

この懸念事項は検証することが困難であるが，率直なところ指導者として Rails の選択を検討するときに最も気になっていた点である．講義や演習時間は有限であるため，基礎と実用を共に満足することは難しい．その状況の中，2018 年度に授業を実施したところ，意外なほどに Rails，Git，Bitbucket，Heroku といったツールを受講生らは自然と使いこなしていた．さらに興味深いのは 3.4 節でも述べたように devise をはじめ，様々な gem ライブラリを“勝手”に探して自分たちのソフトウェアに組み込んで使っていたことである．これまでに技術力もやる気もある一部の受講生がこうした取り組みをすることはあったが，表 3 に示す通り，ほとんどのチームが何かしらの gem ライブラリを採用したという事実は驚きであった．Java にも様々な有用なライブラリがインターネット上に公開されているが，Java EE の演習では学生らの要求に応じてライブラリを紹介することはあったが，勝手に探して取り入れることはなかった．4.1 節でも触れたように，Rails を利用することで同じコード量でも実装できる機能数が増えるため，このことは受講生のやる気に繋がる可能性が考えられ，受講生が進んで情報収集するようになれば「さらに知りたい」という動機を引き出すことが期待できるかもしれない．この懸念についてはさらに検証が必要である．

4.2 授業評価アンケート

本大学では全学部で共通の形式の授業アンケートを期末に実施している．受講生自身の授業態度に関する項目を抜き出してまとめたものを表 5 に示す．2018 年度と 2016 年度でほとんど差がない結果となった．設問 13 については，Cloud9 を採用したため自宅でも環境構築せずに取り組めるようになったためと考えられる．設問 14 は科目の総合評価項目となっているが，カリキュラム改定前後で同じ

©2019 Information Processing Society of Japan

値となっている．想像の域を出ないが，チーム開発では各チームでそれなりの時間を費やして共同作業を進める．ツールや環境が変化することより，共同で作り上げた達成感の方が大きいのではないかと考える．

表 5　授業評価アンケートの比較

設問番号	アンケート項目	2018年	2016年
6	授業に刺激され授業内容に興味を持つようになりましたか。	4.3	4.3
12	あなたは授業中に集中し、私語や授業に関係のないことをしないように心がけましたか。	4.1	4.0
13	あなたはこの授業に対して、1週間で平均何時間、自学自習していますか。	3.0	2.8
14	この教員の授業を10点法で評価してください。	8.3	8.3
15	授業で使用した教室の設備・環境は良かったですか。	4.3	4.3

5. まとめ

本稿では, 本学科の 3 年次後期を対象とした Ruby on Rails による Web アプリケーションのチーム開発の授業計画と 2018 年度の授業実施結果について報告した．結果的に，受講生らは Rails, Git, Bitbucket などのツールを使いこなし，チームで企画した Web アプリケーションを完成させることができた．また，完成させた Web アプリケーションは，カリキュラム改定前の Java EE によるチーム開発のコード量と同程度であり，Rails の支援を受けてより多くの画面数を持つアプリケーションを開発できたことがわかった．

Rails はプロフェッショナルが利用する Web アプリケーションフレームワークではあるが，初学者でも基礎を学びそれなりの時間をかければ，普段，我々が利用しているサービスのような Web アプリケーションを作り上げることができ，達成感を得ることができる．2010 年にフィンランドで始まった世界的なムーブメントである Rails Girls の取り組みの方法論として Ruby＋Rails が選択された理由には同じような背景があったものと推察する[12].

本稿で取り上げた講義・演習科目は来年度も引き続き開講するため，演習中にエラーメッセージなど収集する仕組みを取り入れるなどして，躓き要因の解析や予防するための支援システムの構築に繋げていきたい．

参考文献

[1] “Eclipse”, https://www.eclipse.org, （参照 2019- 4-18）.
[2] “Cloud9”, https://c9.io, （参照 2019- 4-18）.
[3] 高橋圭一, Ruby on Rails による Web アプリ開発の授業実践, 情報処理学会九州支部火の国シンポジウム 2018, 2019.
[4] “Git” , https://git-scm.com, （参照 2019- 4-18）.
[5] “Bitbucket”, https://bitbucket.org, （参照 2019- 4-18）.
[6] “Heroku”, https://heroku.com, （参照 2019- 4-18）.
[7] “JSP Model2”, https://en.wikipedia.org/wiki/JSP_model_2_architecture, （参照 2019- 4-18）.
[8] “Ruby on Rails Tutorial”, https://www.railstutorial.org/book, （参照 2019-4-18）.
[9] “AWS Educate”, https://aws.amazon.com/jp/education/awseducate/, （参照 2019-4-18）.
[10] “ソフトウェア開発データ白書 2018-2019 情報通信業編”, https://www.ipa.go.jp/sec/publish/tn16-005.html, （参照 2019-4-18）.
[11] “RubyGems”, https://rubygems.org/, （参照 2019-4-18）.
[12] 鳥井雪, プログラミング言語 Ruby の最新動向：[Ruby の広がり] 8. Rails Girls とその背景, 情報処理, Vol. 56, No. 12, pp.1184-1186, 2015/11

©2019 Information Processing Society of Japan