Gitを利用した学習管理システムの開発

関西学院大学理工学部
情報科学科　西谷研究室
27013536
高橋駿

2019年3月

概 要

西谷研究室の大津が開発している ruby_learner は，プログラミング言語 Ruby の初学者向けに開発された，Ruby 学習アプリケーションである．このアプリケーションには学習者の進捗状況を管理する機能が備わっておらず，試験運用の際にも，継続的な学習が行えていなかった．

そこで，本研究では ruby_learner の学習状況の管理を行うアプリケーション ruby_learner_checker の開発を行い，グループ学習における学習記録の管理，および学習者の学習意欲の向上を目指す．

本研究で開発したアプリケーションは，複数の学習者が ruby_learner で行なった学習のデータの収集，表示，比較の機能を備えており，目的の一つである ruby_learner で行なった学習の記録の管理を可能にしている．

一方で，もう一つの目的である学習者の学習意欲の向上については，操作性やビューのレイアウトなどが与える印象の点からアプローチを試みたが，いずれも十分な結果が得られたとは言えず，継続的な改良が求められる．

目 次

第1章　はじめに

西谷研究室の大津が開発している，ruby_learner [1] というアプリケーションがある．これは，プログラミング言語 Ruby の学習を CUI 上で行えるアプリケーションである．

このアプリケーションの試験運用が 2018 年度の春学期に西谷研究室の卒業研究生を対象にして行われたのだが，春学期中に ruby_learner の全学習課程を終えた学生は一人もいなかった．

ruby_learner の課程は Ruby の規則や構文等の基本的な知識を身につける設問で構成されており，課程が全ての学生にとって達成不可能なほどに困難であったとは考えにくい．

そこで他に原因がないか調べたところ，ruby_learner には学習の記録を管理する機能がなく，学習の進捗を管理できていなかったことが学習課程を終えられなかった原因ではないかと考えた．

本研究の目的は，西谷研究室内における ruby_learner の学習の記録の管理，ならびに学習者の ruby_learner に対する学習意欲の向上を助けるアプリケーション ruby_learner_checker の開発を行い，プログラミング言語 Ruby の早期習熟を助けることである．

第2章　開発目標

この章では，ruby_learner_checker に求められる要件と，その実装方法について記す.

2.1　実装した機能

1 章で述べたとおり，ruby_learner_checker の開発の目的は

- 学習記録の管理
- 学習意欲の向上

の 2 点である.
これらの目的を達成するために，以下のような機能が必要であると考えた.

- 学習データの集積
- データから得られる学習の成果の表示
- ユーザフレンドリーなインタフェース

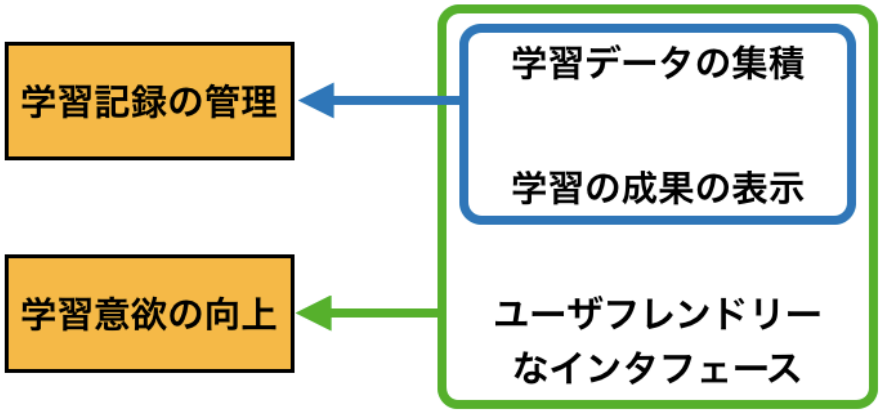

図 2.1: 目的とそのための実装の略図.

それぞれの機能に関する考察と，具体的な実装方法を以下に記す.

2.1.1　学習データの集積

効率的に管理を行うには，管理を行う対象を一箇所に集めるのが有効である．

本研究においても，より効率的な管理を行うために，学習者のデータを一元的に集めるべきだと考えた．

具体的な手法として，Git によるデータ管理を行った．

Git は分散型のバージョン管理システムであり，各ユーザがワーキングディレクトリに全履歴を含んだリポジトリ（データを一元的に保管しておく場所）の完全な複製を持つことができる．

これを用いることで，各学習者のデータの変更を一元的に管理し，またその変更を同期したデータを各学習者のワーキングディレクトリに保持することが可能になった．

2.1.2　データから得られる学習の成果の表示

より効果的な管理を行うためには，データを収集するだけではなく，収集したデータから有意な結果を得る必要がある．

例えば図や表は，単純なデータのリストと比較して視覚的にデータを捉えやすく，有意な結果を得るための理解が非常に容易である．

本研究においても，個人の学習の成果や，他者との学習状況の比較が可能なテーブルやチャートを作成し，学習の成果を分かりやすく表示した．

具体的な手法としては，HTML と Javascript によるビューの作成を行った．

この実装により，ブラウザ上で動作可能なビューを作成し，チャートやテーブルをビュー上で表示することが可能になった．

2.1.3　ユーザフレンドリーなインタフェース

ユーザフレンドリーとは，ユーザインターフェイスの設計思想のひとつで、コンピューターなどが使いやすい状態のこと [2] を指す．

一般的にユーザフレンドリーなインタフェースには，

- 操作が直感的で扱いやすい
- 表示される情報が分かりやすい

などの特徴が挙げられる．

学習意欲を向上させるには，いかに学習者に継続的に学習を行わせられるかが重要であり，またそのためにどのような報酬を学習者に与えるかが問題となる．

そこで，ruby_learner で行った学習の成果をより分かりやすい形で学習者に伝え，学習の成果が身についていることを実感させることで，学習意欲の向上を狙った．

また，そのための操作を簡潔に行えるようにすることで，継続的な利用の際に煩わしさを与えないようにした．

具体的な実装として，前者にはCSSやChart.jsを使用し，より視覚的に情報を得やすいビューを作成し，また見栄えを良くすることで使用感の向上を狙った．

後者にはThorによるコマンドの実装により，簡潔でかつ行った操作の意味をつかみやすい操作をユーザに提供できると考えた．

第3章　開発に利用したライブラリ等

ruby_learner_checker の開発にあたり，以下の既存のシステム，ライブラリ等を使用した．

3.1　言語

開発に用いたプログラミング言語は主に Ruby であり，ビューの実装の一部に HTML と Javascript を使用した．

3.2　GNU Emacs

GNU Emacs は Emacs テキストエディタの一種であり,「今日利用できる最もパワフルなテキストエディタ」と称されている [3, 訳] . 本研究においては，後述する Org-mode が標準リリースに含まれている GNU Emacs26.1 を使用している．

3.3　Org-mode

Org-mode は，Emacs の拡張ライブラリの一種であり，高速で効率的なプレインテキストのシステムを使ったファイルの編集が行える．本研究においては，Org-mode の機能の一つである HTML への出力機能を使用している．

3.4　Git

Git は，プログラムのソースコードなどの変更履歴を記録・追跡するための分散型バージョン管理システムである [4] .

本研究においては，各学習者の学習データを集積するために機能の一部を用いる．

3.5　Chart.js

Chart.js は，デザイナーと開発者のための，簡潔で柔軟に Javascript でのチャート作成が行える [5, 訳] ライブラリである．

3.6 RubyGems

RubyGems は，Ruby 言語用のパッケージ管理システムであり、Ruby のプログラムと（gem と呼ばれる）ライブラリの配布用標準フォーマットを提供している [6] .

本研究において使用した RubyGems のライブラリを以下に記す.

3.6.1 Thor

Thor は、コマンドラインツールの作成を支援するライブラリであり，Git や bundler のようにサブコマンドを含むコマンドラインツールを簡単に作成することができる [7] .

3.6.2 Rack

Rack は，指定したファイルを独自の Ruby DSL として読み込み，DSL で指定した様々なミドルウェア，アプリケーションを組み合わせて Web サーバを立ち上げることができる rackup というコマンドを提供するライブラリである [8] .

本研究においては，ビューを表示するための localhost でのサーバ立ち上げに用いる.

3.7 CSS

CSS (Cascading Style Sheets) は，ウェブページのデザインやレイアウトなどのスタイルを指定するための言語である.

本研究では，fniessen 氏がオープンソースで提供している，ReadTheOrg [9] を使用している.

第4章　ruby_learner_checkerの操作法

この章では，ruby_learner_checker の具体的な使用方法について記す．

4.1　インストール

ruby_learner_checker は，西谷研究室内で Ruby の学習を行う際のデータの管理を主な目的として開発されたため，西谷研究室で利用されている GitHub から clone する必要がある．

```
$ git clone git@github.com:TeamNishitani/ruby_learner_checker.git
```

上記のコマンドを実行することで，カレントディレクトリ内に ruby_learner_checker ディレクトリのクローンが作成される．

4.2　初期設定

4.2.1　gem ライブラリのインストール

ruby_learner_checker には，RubyGems で公開されているライブラリを使用している．
ruby_learner_checker の実行に必要なライブラリをインストールするには，カレントディレクトリを ruby_learner_checker に設定し，以下のコマンドを実行する．

```
$ bundle install
```

4.2.2　リポジトリの設定

ruby_learner_checker では，学習データの管理のために Git を使用する．そのため，運用に際して，始めにリポジトリを設定する必要がある．

- リモートリポジトリの設定

リモートリポジトリは，ホスト上に配置して複数人で共有するためのリポジトリである．
リモートリポジトリを作成するには，データを集積するホストとなるマシン内で，以下のコマンドを入力する．

```
$ git init –bare –config path/to/members.git
```

これで，指定したパスのディレクトリにリモートリポジトリが作成される．

- ローカルリポジトリの設定

ローカルリポジトリは，ユーザ一人ひとりが利用するために、自分の手元のマシン上に配置するリポジトリである．

ローカルリポジトリを作成するには，ruby_learner_checker を運用するマシン内で，以下のコマンドを入力する．

```
$ cd ruby_learner_checker/storage $ git clone host@address/path/to/members.git
```

これで，ruby_learner_checker の storage ディレクトリにローカルリポジトリが作成される．

4.3 動作方法

ruby_learner_checker の操作は基本的に CUI 上で用意されたコマンドを実行することで行う．

また，全てのコマンドはカレントディレクトリを ruby_learner_checker に設定して行う．

4.3.1 基本動作

基本的な動作となる学習データの更新は，以下のコマンドを順に実行することで行う．

- sync

```
$ bundle exec exe/ruby_learner_checker sync [*args]
```

ruby_learner の最新の学習データを取得し，リモートリポジトリと同期を行う．

引数に学習者名を文字列で取る．複数の引数を入力することで，ruby_learner に実装されているペアプログラミングモード等の，一つのマシンで複数の学習者が同時に学習を行う場合にも対応している．

具体的な内部動作は下記の通りになっている．

1. ローカルリポジトリを最新のリモートリポジトリの状態と同期し，
2. 学習データに受け取った学習者名を付与して
3. リモートリポジトリに送信する．

また，初回の sync コマンド実行時の一つ目の引数はデフォルトの学習者名として記録され，２回目以降は引数を入力しなかった場合，その学習者名が付与される．

- merge

```
$ bundle exec exe/ruby_learner_checker merge
```

ローカルリポジトリの学習データから，ビューを生成する．
具体的な内部動作は下記の通りになっている．

1. 全ての学習データを学習者名ごとに仕分け，
2. チャートを表示するための csv，html を生成するための org を作成し，
3. org から html を作成する．

- view

```
$ bundle exec exe/ruby_learner_checker view
```

ローカルホストを立ち上げ，merge コマンドの実行により作成されたビューを表示する．

4.3.2 その他の動作

主要な動作に含まれないが，ruby_learner_checker に実装されているコマンドと，その動作内容を以下に記す．

- pull

```
$ bundle exec exe/ruby_learner_checker pull
```

前回の更新からローカルリポジトリの内容に変更がない場合に，ローカルリポジトリを最新のリモートリポジトリの状態と同期したい際に使用する．

- name

```
$ bundle exec exe/ruby_learner_checker name [arg]
```

sync コマンドに引数を入力しなかった場合に，自動で学習データに付与される学習者名を変更する際に使用する．
引数に取った文字列を，新しいデフォルトの学習者名として設定する．

4.4　ビュー

ビューは以下の 3 種の html から構成される．

4.4.1　portal

ビューを立ち上げた際にはじめに表示されるページ．ブラウザ上では図 4.1のように表示される．

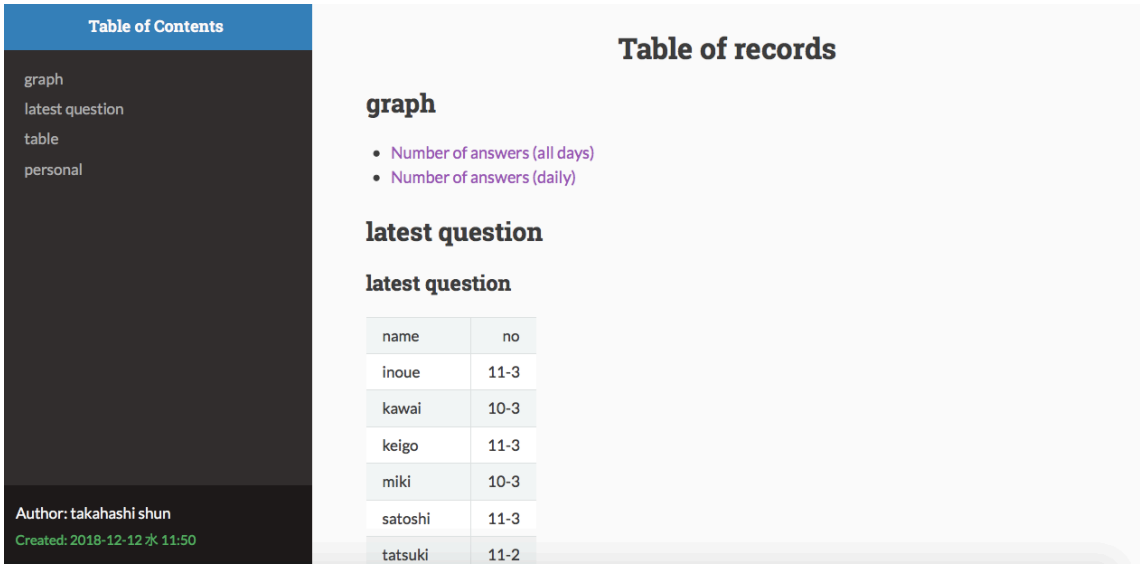

図 4.1: portal.html の表示例．

portal のページは，以下の要素で構成される．

- チャートを表示するページへのリンク
- 各学習者の最新の単元，問題の番号
- 各学習者の全単元の成否表
- 各学習者の personal ページへのリンク

この内，成否表は図 4.2のように表示される．

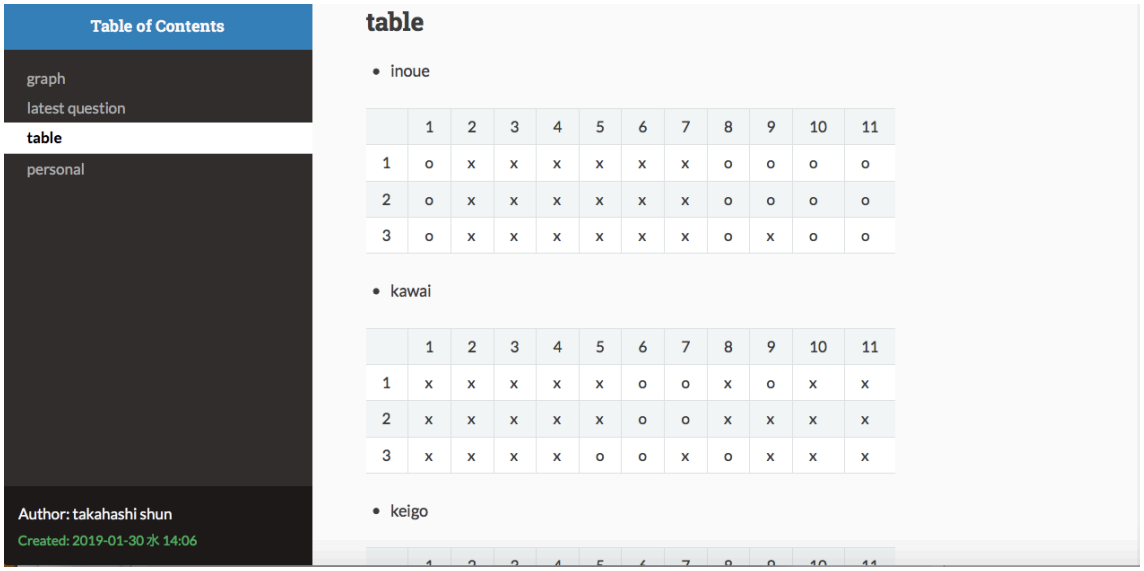

図 4.2: portal.html 中の成否表の表示例.

ruby_learner の課題は 11 個の単元に，それぞれ 3 個の設問が用意されており，表の横列が単元の番号，縦列が設問の番号に対応し，その設問を履修したことがあるかどうかを表示する.

4.4.2 chart

merge メソッドで作成されたデータから，学習状況に関するチャートを表示するページ.
表示されるチャートは以下の 4 つである.

- Bar chart

学習者名と，その累計の学習回数，平均の学習回数を棒チャートで表示する.
横軸に学習者名を取り，縦軸にその学習者の累計の学習回数をとっている．また，赤い線は全学習者の学習回数の平均を表示している.
実際の表示は図 4.3のようになる.

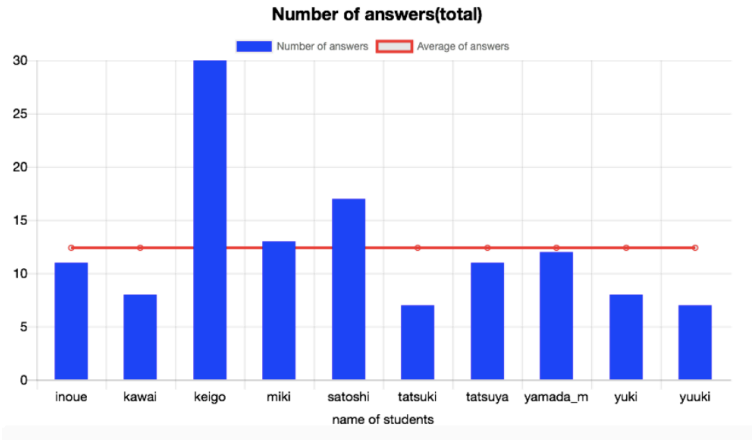

図 4.3: name_count.html の表示例.

- Polar chart

学習者名と，その累計の学習回数を鶏頭チャートで表示する．
　上部に学習者名とチャート内で対応する色を表示し，下部に累計の学習回数のチャートを表示している．
　実際の表示は図 4.4のようになる．

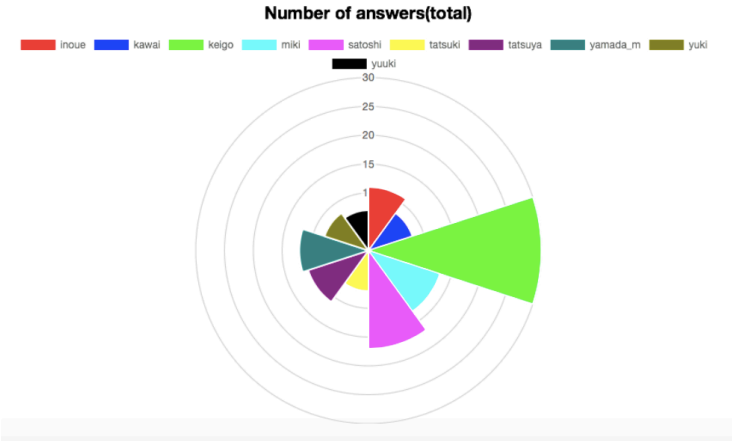

図 4.4: name_count_polar.html の表示例.

- Line chart (Daily)

学習者名と，その日毎の学習回数を線チャートで表示する．
　実際の表示は図 4.5のようになる．

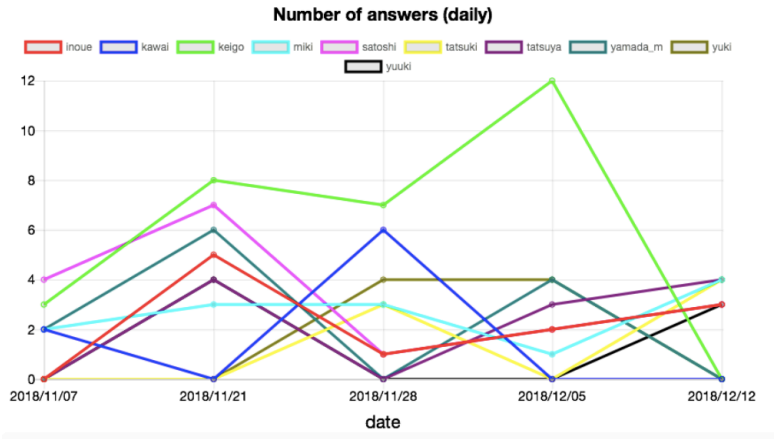

図 4.5: date_count.html の表示例.

- Line chart (Sum)

学習者名と，その日毎の累計学習回数を線チャートで表示する.
実際の表示は図 4.6のようになる.

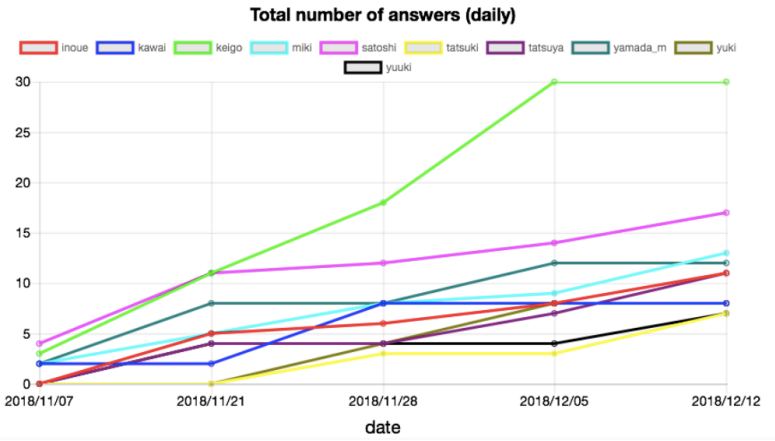

図 4.6: date_sum.html の表示例.

4.4.3 personal

各学習者の学習データを表示するページ.
1 ページに 1 人の学習者が割り当てられており,

学習者名, 年月日, 終了時刻, 学習時間, 単元

を 1 行とするテーブルを表示する.
実際の表示は図 4.7のようになる.

図 4.7: 個人の学習記録の表示例.

第5章　実装過程における考察

この章では，ruby_learner_checker の開発を進める過程で，特に大きな変更点や決定を行った事柄，またその際に行った考察を記す．

5.1　CUI ベースのアプリへ

当初，本研究は Ruby on Rails を用いたデータの集積，管理を行うアプリケーションの開発も検討されていたが，
実際のアプリケーションを運用する際の規模が個人から複数名でのグループワークという小規模な範囲で収まることが決定し，アプリケーションに要求される仕様が

- 精々十数人分程度の学習データしか扱わない．
- ネットワークアプリケーションである必要がない．

ことから，Ruby on Rails および SQL で扱うにはデータ量の規模が小さく，個人のマシン内でプレインテキストのデータとして扱っても十分に管理が可能であると判断した．
また，ruby_learner が CUI 上のみで動作するアプリケーションであるため，動作環境が統一されている方が好ましいと判断し，基本的な動作を全て CUI 上で行えるように開発を進めた．

5.2　わかりやすいコードを

実際のプログラミングの現場では，一度書いた何らかのコードの中身を二度と見ないということはまずありえない．また，自身が書いたコードの中身を他者が見るということも珍しくない．
「動作する綺麗なコード」を書くことは，プログラマーにとって最も重要な能力の一つである．Kent Beck は自身の著書で動作する綺麗なコードについて以下のように述べている [10, 前書き p.9] ．

> 動作する綺麗なコードはあらゆる意味で価値がある．
>
> - 開発が予測可能になる．完成したかどうかがわかり，バグが残っているかを心配する必要もない．

- コードが伝えようとしていることを余すところなく受け取れる．最初に思いついたコードを書き殴っただけで終わりなら，再考してより良いコードを書くチャンスは永遠に来ない．
- あなたが作るソフトウェアのユーザを快適にする．
- チームメイトはあなたを信頼し，あなたもまたチームメイトを信頼する．
- 書いていて気持ちがいい．

本研究においても，今後他者がruby_learner_checkerを改良する際の可読性の向上，また自身のプログラミング能力の向上を目的として,「動作するコード」から「動作する綺麗なコード」へのリファクタリングを行った．

5.2.1 メソッドの抽出

チャートを描画するのに必要なcsvファイルは，当初は全てmk_csvメソッドという一つのメソッドの中で出力されていた．

この時，同一メソッド内で同名の一時変数が初期化されて使い回されており，混乱を招く恐れがあったため，メソッドの抽出 [11, p.128] を行い，一つのcsvファイルを出力する行程を一つのメソッドに分割し，同一メソッド内での一時変数の使い回しが行われないようにした．

5.2.2 不要な行程の削除

コードの実行過程において，余分な中間ファイルを生成することはディレクトリ構造の複雑化を招き，プログラムから逐次ファイルを読み込まなければならず，コード自体の冗長化にもなる．

ruby_learner_checkerの開発においても，実行過程でstu_list.csvという中間ファイルを出力していた．

stu_list.csvは，収集したデータから抽出した全ての学習者名を各行ごとに記述しているだけの中間ファイルであり，ファイルを読み込み，学習者名を配列に格納する際にしか利用されていなかった．

そこで，学習者名を格納した配列をクラス変数にすることで，クラス内のどこからでも配列を参照できるようにし，ファイルを出力する行程を省略した．

5.2.3 クラスの抽出

開発当初の設計では，図5.1の構造で，ruby_learner_checker.rb内のRubyLearnerCheckerクラスに全ての動作を行うコードが記述されていた．

この構造では，いざコードの実装を確認しようとした時に，非常に長く煩雑なコードの中から目的のコードを探し出すまでに時間がかかってしまう．

これを，Martin Fowler が記したリファクタリングのカタログの内，クラスの抽出 [11, p.199] に則り，図 5.2 の構造の通り，ruby_learner_checker のそれぞれのコマンドが行う動作を別々のファイルに分け，クラスとして記述し，コード間の関連性を明確にした．

また，以前は exe ファイルに実装されていたコマンドの処理を ruby_learner_checker.rb 内に移し変え，exe の記述はそれを読み込むだけに変更することで，全ての動作を行うコードを一箇所にまとめて置くことができるようにした．

この構造にしたことで，目的のコードがどのコマンドの動作に含まれているかわかっていれば，コードを探す手間を削減することができた．

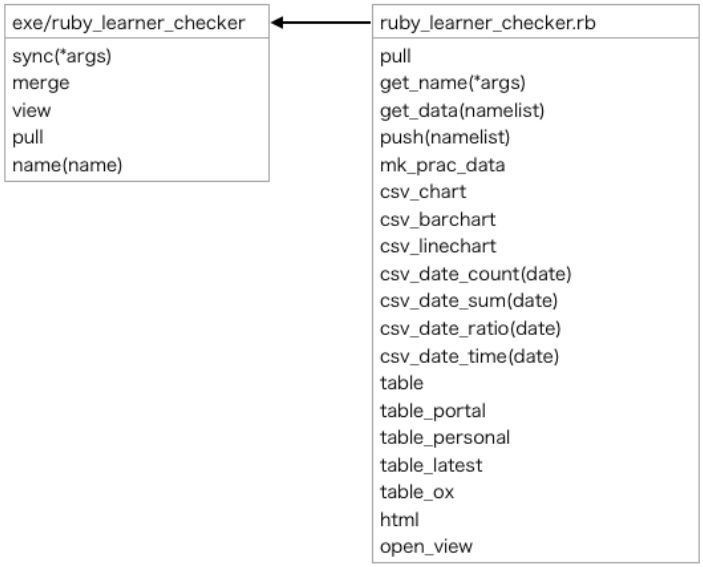

図 5.1: 変更前のコードの構造．

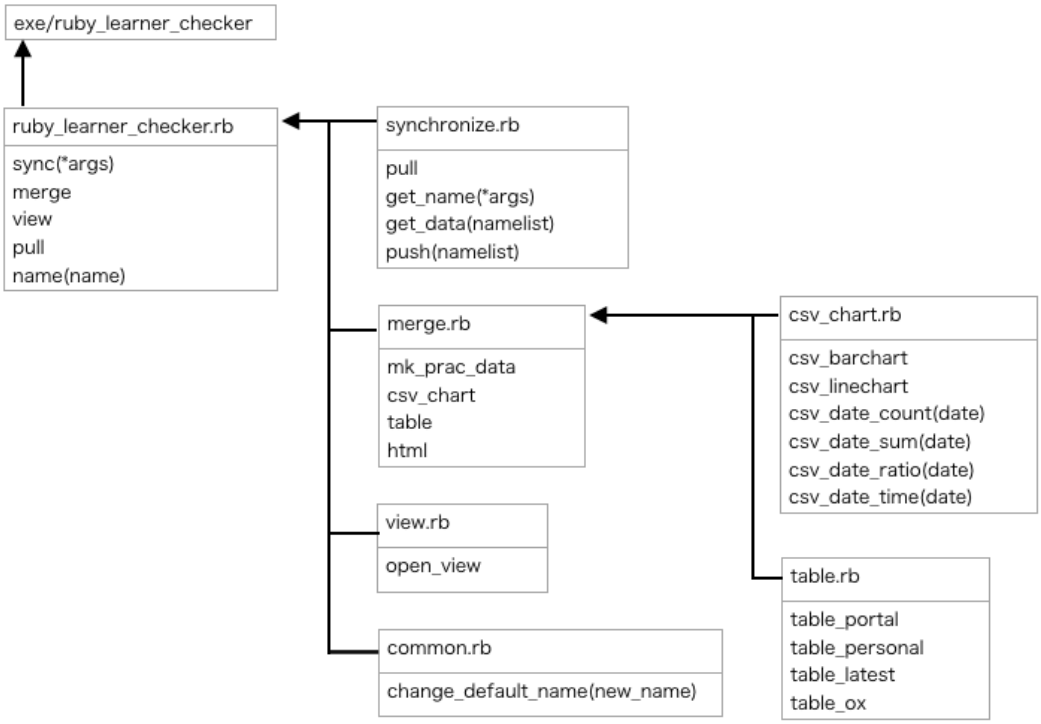

図 5.2: 変更後のコードの構造.

5.3 rsync から Git へ

当初，学習者の学習データを収集するために，UNIX 系の OS で動作する，遠隔地間のファイルやディレクトリの同期を行うアプリケーションソフトウェアである rsync を用いていたが，データの管理に Git を使わないのかという意見を受けた.

そこで，Git と rsync の比較を行い，どちらを利用するべきか検証を行った.

5.3.1 Git,rsync の比較

コマンドライン上での動作 どちらも問題なく動作する．Git は GUI で利用するためのサポートが充実しているが，本研究では使用しないため割愛する.

ログの管理 Git 自体にログを管理する機能 (git log コマンド) があるため,「誰が」「いつ」データの更新を行ったのかがわかりやすい．rsync はディレクトリやファイルを比較し，同期を行うだけなので，ファイルに更新日等の記録は残るが，rsync のみでの管理は困難である.

ssh 通信の可否 Git，rsync のどちらも，ssh 鍵の設定をすればパスワードの入力なしに ssh 通信が利用可能である.

データの保全性 rsync は同期以前のデータを保持しておくことができないため，手動でバックアップを取る必要がある．Git は記録しているログから，リポジトリの状態を復元することができる．

運用の簡易性 rsync は同期したい 2 つのディレクトリ間で処理が完結する．コマンドの記述法もコピー（cp コマンド）と似ており，その延長で使用できる．Git は利用する際にリポジトリを設定しておく必要があるため，リポジトリを運用するための知識が必要になる．

以上の点をまとめると，表 5.1と書ける．この表から，rsync は個人での簡単なデータの管理，単発的なタスクに向き，Git は複数人で行う作業，継続的なタスクにおける管理に向いていると言える．

	rsync	git
CUI	○	○
ログ管理	×	○
保全性	×	○
簡易性	○	△

表 5.1: Git と rsync の比較表．

本研究では複数人の学習者のデータを一元的に管理する必要があり，ログを管理できる点が学習管理に有効であると考え，rsync で行っていたデータの管理を Git で行うように変更した．

5.3.2 sync コマンドへの統合

データの管理に rsync を利用していた時点では，データの集積，管理を push，pull コマンドで行っていた．

```
$ bundle exec exe/ruby_learner_checker push [*name]
$ bundle exec exe/ruby_learner_checker pull
```

rsync から Git への変更に際して，データの管理を行うために必要な手順が変わったので，2 つのコマンドを統合し，新たに sync コマンドを作成した．

```
$ bundle exec exe/ruby_learner_checker sync [*name]
```

これにより，ruby_learner_checker の操作に必要なコマンド数が減り，実行するコードをまとめて記述することでコードの管理が容易になった．

5.3.3　コマンドの比較

push,pull コマンドとの主な変更点として，リモート環境との同期を行う際の処理の順番を変更した．

rsync を利用していた際は，

1. ローカル環境で自身の学習データを更新してから，
2. rsync でリモート環境にその変更を同期し，
3. また rsync を使ってリモート環境から他の学習者の記録をローカル環境に同期する．

という手順をとっていた [図 5.3]．

Git でこの手順を取ると，リモートリポジトリに変更を同期する際に，自身のリポジトリで変更を加えたファイルに別のリポジトリから別の変更が加えられていた場合，conflict を起こし，予期せぬデータの消失を招く危険性がある．

そのため，先に pull を行なってローカルリポジトリにリモートリポジトリの更新状況を同期することで，conflict を起こさないような手順をとっている [図 5.4]．

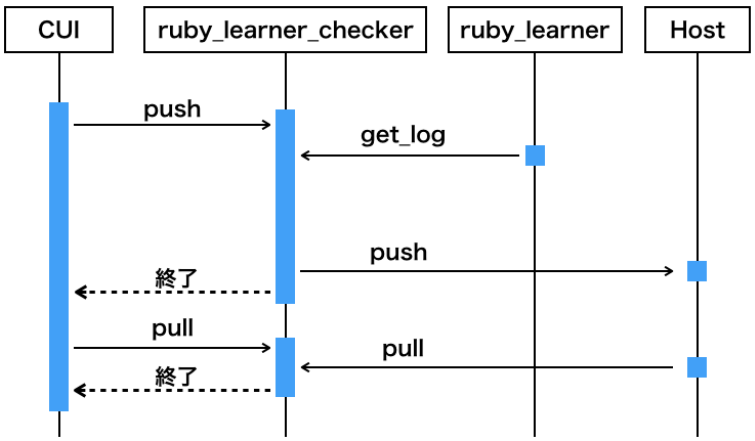

図 5.3: push,pull コマンドのシーケンス図．

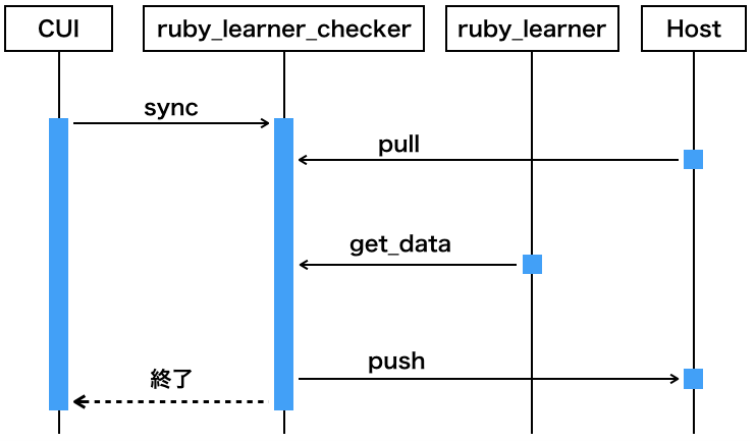

図 5.4: sync コマンドのシーケンス図．

5.4 viewの調整

学習記録を管理するアプリケーションには，収集した記録から学習の進捗や成果がすぐにわかるような機能がついていることが求められる．

個人の指標 自分の行ってきた学習がどの程度の成果なのか，目標に到達するにはあとどれくらいの学習をすべきなのかといった目安を与える．

学習者間の指標 他者と学習記録を比較できるようにすることで，学習者間での競争心や互いに学習状況を見られているという意識を持たせる．

自身の学習の成果や他者と比較した際の差がより伝わりやすい形で提供することで，進捗管理をしやすくすると共に，これらの指標とその効果をより大きく与えられると考えた．

そこで本研究においても，学習記録を分かりやすく表示するためのチャート作成を行った．

5.4.1 評価方法

より学習管理に効果的なチャートを作成していく過程で，実際にユーザがアプリケーションを使用した際の印象を調査するため，複数のチャートのモデルを作成し，秋学期の領域実習生と卒業研究生を対象にそれぞれのモデルに対して学習管理の観点から学習に関して効果的な印象を受けるかというアンケートを行った．

対象としたチャートのモデルは以下の 6 種である．

- 累計学習回数の棒チャート [図 4.3]
- 累計学習回数の鶏頭チャート [図 4.4]
- 日毎の学習回数の線チャート [図 4.5]
- 日毎の累計学習回数の線チャート [図 4.6]
- 日毎の ruby_learner の全単元の学習達成率の線チャート [図 5.5]
- 日毎の学習時間の線チャート [図 5.6]

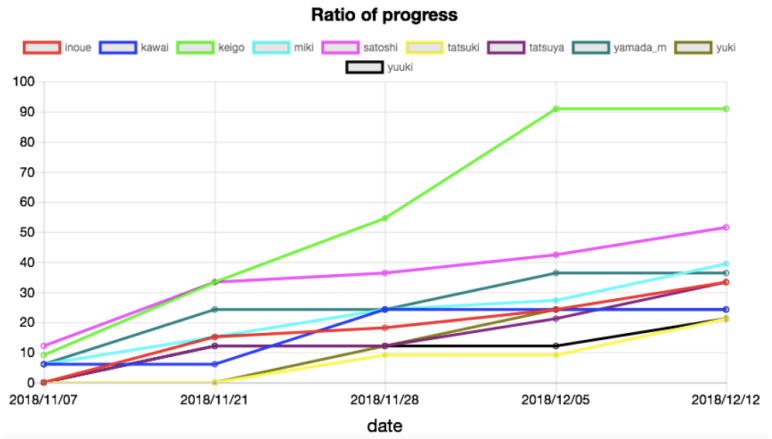

図 5.5: date_ratio.html の表示例.

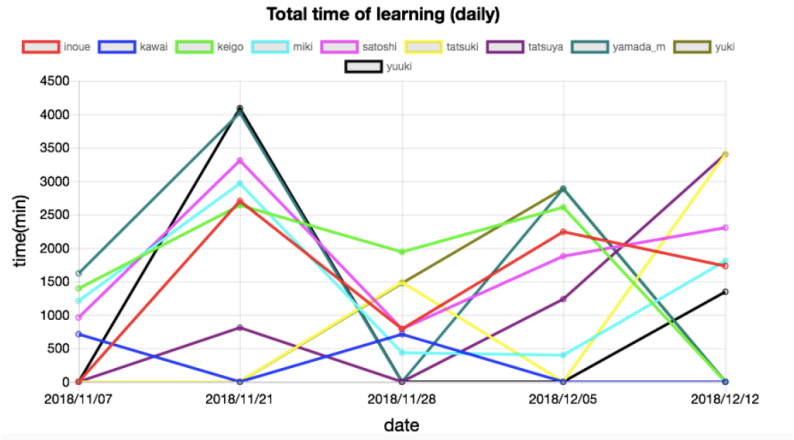

図 5.6: date_time.html の表示例.

アンケートの結果，日毎の累計学習回数の線チャート, 累計学習回数の鶏頭チャートに対する効果的な印象が強く，逆に，学習達成率の線チャート, 日毎の学習時間の線チャートに対する印象は良くないという結果が得られた.

5.4.2 考察

アンケートから得られた具体的な意見と，それから得られた考察を以下に記す.

- 日毎の学習回数と日毎の累計学習回数の比較

日毎の学習回数のチャートに対して，日毎の累計学習回数のチャートがより好印象である点について，長期的な学習量の比較が行いやすいという点が，日別の学習量の比較を行えるという点よりも良い印象を受けるという意見が多かった.

- 棒チャートと鶏頭チャートの比較

累計学習回数を表示するチャートについて，棒チャートより鶏頭チャートがより好印象である点については，棒チャートに対し，鶏頭チャートの方が目新しく，強いインパクトを受けるという意見が大半であった．高さを比べる棒チャートよりも扇状に面積が大きくなっていく鶏頭チャートの方が，比較した際の差を大きく感じられ，強い印象を与えたと考えられる．

- 学習達成率のチャート

学習達成率のチャートについては，同単元を繰り返し学習しない場合は日毎の累計学習回数のチャートと形状が類似し，ruby_learner 全体の進捗を計測するという用途も，portal.html に表示される全単元の成否表と重複するために独自の必要性を見出されなかったものと考えられる．

- 学習時間のチャート

日毎の学習時間のチャートは，領域実習の講義時間内での試験運用という環境で，学習者間の学習時間という情報にあまり意味がなかったために，印象づけられなかったものと考えられる．

ただ，本研究においては検証できなかったが，各自が自学自習を行える環境での運用においては，学習時間のチャート化は十分に有用性が認められる可能性がある．

第6章　まとめ

本研究を通して，行った開発等を以下に記す.

- 複数台のマシンからのデータの収集.
- 学習データを元にしたチャートの生成，表示.
- CUI 上で動作するコマンドの実装.
- 効果的なチャートに関するアンケート.
- 可読性の高いクラス構造.

これらの事柄から，以下のような成果が得られた.

- 一元的なデータの管理.
- 学習の成果の視覚化.
- ユーザに要求する操作数の削減.
- 学習意欲の向上につながるチャートの傾向.
- 第三者によるコードレビューが行いやすい.

開発されたアプリケーション ruby_learner_checker は，Git による学習データの収集，ビューの生成による学習データの視覚化を行うコードの実装により，本研究の目的の一つであった ruby_learner で行った学習の進捗の管理が可能になった.

しかし，もう一つの目的である学習意欲の向上については，インパクトの強いレイアウト，他者との比較がしやすいチャートが有効である可能性が得られたが，確実な実証，実装には至らなかった.

そこで，本研究内で達成できなかった課題の解決への一案として，アプリケーション全体のコードの可読性が高くなるような実装を行い，継続的な改良を加えやすくしている.

参考文献

[1] 大津隆輝, ruby_learner 1.2.9, https://rubygems.org/gems/ruby_learner/versions/1.2.9 (accessed on 29 Jan 2019).

[2] ユーザーフレンドリー (ゆーざーふれんどりー) とは - コトバンク, https://kotobank.jp/word/ユーザーフレンドリー-9485 (accessed on 20 Feb 2019).

[3] "'Learning GNU Emacs, Third Edition': A Guide to the World's Most Extensible, Customizable Editor", https://www.oreilly.com/pub/pr/1285 (accessed on 30 Jan 2019).

[4] Git - ウィキペディア, https://ja.wikipedia.org/wiki/Git (accessd on 4 Feb 2019).

[5] Chart.js | Open source HTML5 Charts for your website, https://www.chartjs.org (accessed on 11 Feb 2019).

[6] RubyGems - ウィキペディア, https://ja.wikipedia.org/wiki/RubyGems (accessed on 4 Feb 2019).

[7] Thor の使い方まとめ - Qiita, https://qiita.com/succi0303/items/32560103190436c9435b (accessed on 4 Feb 2019).

[8] Rack とは何か - Qiita, https://qiita.com/k0kubun/items/248395f68164b52aec4a (accessed on 7 Feb 2019).

[9] GitHub - fniessen/org-html-themes: How to export Org mode files into awesome HTML in 2 minutes, https://github.com/fniessen/org-html-themes (accessed on 7 Feb 2019).

[10] Kent Beck (著), 和田 卓人 (翻訳), テスト駆動開発, (オーム社, 2017/10/14).

[11] Jay Fields (著), Shane Harvie (著), Martin Fowler (著), Kent Beck (著), 長尾 高弘 (翻訳), リファクタリング:Ruby エディション, (アスキー・メディアワークス, 2010/2/27).

謝辞

本研究を進めるにあたり，多大な御指導，御鞭撻をいただいた西谷滋人教授に，厚く感謝を申し上げます．また，数多くの助言，助力をいただいた西谷研究室の同輩の皆様，西谷研究室の領域実習生の皆様にも感謝の意を表します．