Jekyllを利用した学習管理システム

関西学院大学理工学部
情報科学科 西谷研究室
27016633
中堀佑紀

2020年3月

概 要

学習する上で最も効率よく学ぶ方法は人から聞くことである．そこで誰が誰から何を学んだかということを研究室内で共有できれば，研究室全体の学習効率があがると考え，また生活習慣を見直し，他者や教示者に指摘されることで学習時間の向上につながると考えた．そこで誰が誰から何を学んだかを表示する"バッジシステムを利用したテーブル表示とテキストのリスト化", 生活習慣改善のための"カレンダーアプリを利用した生活記録のチャート化"を行った．

バッジのテーブル化によって, 誰が何を学習したのか一目で分かるようになり，それを見ることで，誰に何を聞けばいいか分かり，テキストのリスト化により次に何を学習すればいいのか把握しやすくなった．またカレンダーアプリを用いた生活の記録を図や表にして日常の記録を残していくことにより．自らの生活を振り返ることができた．この2つを利用することで，教示者が課題を与えられ，達成できれば次の内容に進み，達成できなかったり，休んでた場合，バッジを所持している者に指導してもらったり，どれだけ学習して課題が達成できた，達成できなかったと教示者が他の受講者の学習量を把握することができる．

しかしまだシステムとして改善点が見られ，今後の課題として，プログラムのテストとリンクしたバッジテーブルを作成することによる作成したプログラムを即座に見直せるようにし，学習効率の向上につなげる．また，カレンダーアプリへの入力が手間なため，スクリーンタイムでのデータをカレンダーアプリへ反映させることで日常記録の入力の手間を減らすことを可能にし，さらに表示させる図や表の数を増やし，細かなところまで変化が分かるようなシステムを作成することがシステムの改善に繋がるのではないかと考える．

目 次

図 目 次

第1章　序論

1.1　背景

徒弟制を模した学習システムの開発を目指して，まずはじめにバッジ集めを利用した学習支援システムとして，GitHub の README.md 等で使われているデジタルバッジ [1][2](以降，バッジと称す) を実装した．バッジの色を学生ごとに色分けし，これをテーブル表示することで誰がどの学生に何を教えたかといった学習管理ができると考え研究を進めた．西谷が開講する大学院の講義にて同研究室生の学習状況を図る予定であった．だが研究室生の参加率は悪く，データをとることが困難になった．そこで学習管理の根本的な原因を調べるため，学生の生活記録を録り，普段の生活から何を行っている時間が多いのか，調べることにした．学生が所持しているスマートフォンやパソコンにインストールされているカレンダーアプリに日々の活動を記録させ，そのデータを元に可視化することで自身の行動を客観視し，自身の生活を改め，学習に向ける姿勢を再考させ，生活習慣の改善または学習の能率化を促し，学習意欲の向上を図ることとした．また学生同士の生活記録を比較することで競争心を芽生えさせることを図った．

1.2　開発目標

1.2.1　認知的徒弟制の確立

学んだ内容を実際に応用することができている学生は少ない．それは学生が授業の中でやることの意味を考えずに，ただ単に授業に参加して，何も考えず暗記して試験に望むからである.「なぜそうなるのか」と学ぶことの本質を理解することなく終えてしまう．それでは本当の力が身につかず，無意味な時間を過ごすこととなってしまう．何かを学ぶには誰かから教えてもらうことが一番手っ取り早い．徒弟制とは「師匠」と「弟子」といった立場が上の者と下の者が存在する関係を築き，立場が下の者が立場が上の者から技術を学んでいく学習方法である．西城卓也はジーンレイブの仕事を次のようにまとめている [3].

伝統的徒弟制度の特徴として，

1. 知識は問題の解決に役立つものである
2. 実にさまざまな文脈の中で知識は獲得される
3. 学習者に与えられる課題は，職場で日々発生する，解決する必要に迫られた問題や，どうしてもしなければならない仕事である

伝統的徒弟制度に対して，段階を踏んで学びが深化する徒弟制となる認知的徒弟制と言う概念が生まれた．ブラウンらによりまとめられた各段階は次に通りである．

1. モデリング：教育者がまず学習者にデモンストレーションを見せる．
2. コーチング：教育者は学習者に実際にその技能を練習させ，その様子を観察しながらフィードバックをする．
3. 足場作り：学習者はさらにさまざまな作業に挑戦する．教育者はその作業の難易度に合わせて足場をつくって手助けしたり，成長に伴って徐々に支援を減らしていく（fading）．
4. 明確化：学びを確実なものにするため，学習者の技術や思考を言語化させるよう教育者は促す．
5. 反映：教育者は，学習者自身のパフォーマンスについて振り返りを促す．
6. 探求：教育者は，次の課題を自主的に探索するよう学習者に考えさせる．

上記に倣い，本研究では，この認知的徒弟制を踏まえてその制度を支援するシステムを作成することを目標とした．バッジシステムによって誰が誰に何を教えたかが分かるように開発を行った．

1.2.2 収集した生活の記録を可視化

各学生の学習管理をするにあたって，まずどのような生活を行っているのかデータを集める必要がある．そのため，管理対象であるデータを収集するにあたって，一箇所にデータを集めることが有効であると考えた．そのため，Github によるデータ管理を行った．データを収集するだけでなく，収集したデータを可視化することで客観的に捉えることができ，新たな考察を得ることができる．データを図や表にすることで，膨大なデータを視覚的に捉えやすく，また容易に比較することを可能にする．そのため，学生ごとの生活記録，また一人の学生の一週間単位の生活記録の比較をさせるため，データを図や表にまとめることを目標とした．その手段として Jekyll という静的サイトにて表示させ，ブラウザ上でチャートやテーブルを表示することで他者との学習状況を比較することが可能となった．

1.2.3 ユーザーフレンドリーな CUI

CUI でできるだけユーザーフレンドリーな設計を心がけた．ユーザーフレンドリーとは，ユーザインターフェイスの設計思想のひとつであり，コンピュータなどの使用者が操作しやすい状態のこと [4] を表す．そのため，ユーザーフレンドリーな設計となれば，GUI での設計を想像する方が多いだろう．GUI はマウスを使って直感的に操作できるため，誰もが簡単に扱える．しかし複雑な操作を行うことにおいては向いていない．それに比べて CUI は文字のみで操作を行うため，簡単に操作することは難しいが，その反面，複雑な作業を行うことにおいては長けている．CUI での操作において shell を用いた開発は

必須である．shell とはプログラマーとしてのまず一歩である．そのため，プログラマーの実践的な状況下を意識し，操作の取得を行うよう開発を行った．CUI でできるだけユーザーフレンドリーなインタフェースの設計に以下の項目を意識した．

- 説明書がなくても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
- ファイルやフォルダが決まった場所にある

そのため作成したプログラムのディレクトリ構造を何が置かれているか一目で理解できるような構造になるよう取り組んだ．また後に説明している Rakefile の扱いについて，通常の引数のとり方とは異なり，コマンドラインでの操作を意識した操作ができるようプログラムの設計を心がける．また操作方法が分からない，忘れてしまったとしても使えるように注意事項を記入した README を作成し，入力例を記述した．データをとってからインターネット上で閲覧できるまでの流れは図 1.1になる．

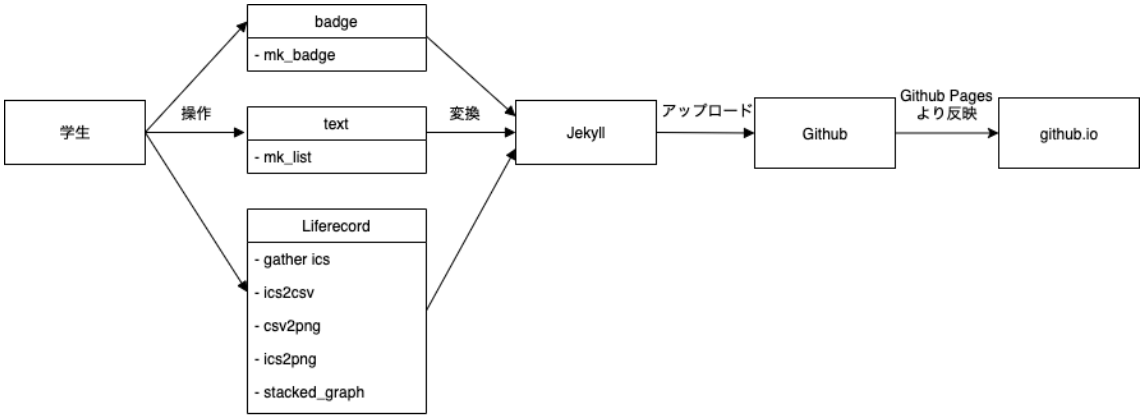

図 1.1: 各入力データがインターネット上で閲覧できるまでの流れ．

第2章　開発環境と手法

2.1　動作について

研究及び学習の足場作りとして作成したプログラムは大きく分けて２つで,

- バッジシステムを利用したテーブル表示とテキストのリスト化
- カレンダーアプリを利用した生活記録のチャート化

である．バッジのテーブル表示を「mk_badges」, テキストのリスト化を「mk_list」と命名したプログラムを作成した．本研究の開発に用いたプログラミング言語は主に Ruby である．またビューの実装の一部に Python を使用した．カレンダーアプリによる学習管理のシステムを「Liferecord」と命名し，複数の処理が必要なため， Rake を用いて作成した．これらのプログラムを実行し，作成したものを Github でデータ収集を行う．Github で収集したデータを Github Pages を用いることで，Jekyll で表示させ使用する流れである．実際にコマンドを実行した図 2.1である.

2.2　RubyGems

RubyGems[9] は Ruby で用いることのできるパッケージを公開することができるサービスである. 公開されたパッケージは

```
$ gem install bundler jekyll
```

といった形でコマンドを入力することで誰でも取得することができる．また取得したパッケージは shell で動作させることが可能である．本研究で使用される Rake や Jekyll も RubyGems より取得し，使用してある．

2.3　Rake

2.3.1　about

複数のコマンド処理を行うため，コマンドが一連であると便利である．そのため，開発システムの中で一連のコマンドの自動実行のため，Rake を用いた．Rake とは，Ruby 公式ドキュメントより以下のように記述されてある [5].

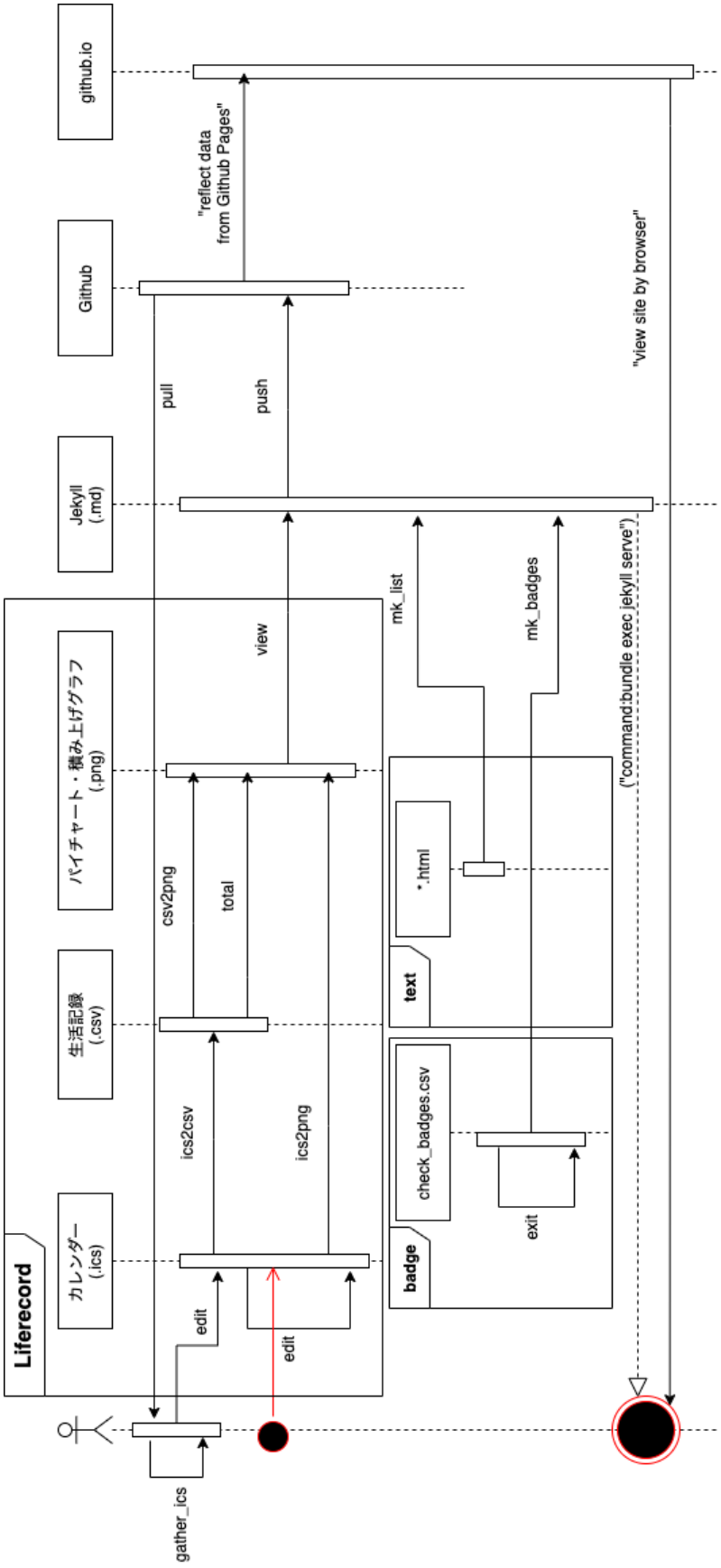

図 2.1: 各コマンドを実行した際の流れ.

Rake は Make によく似た機能を持つ Ruby で書かれたシンプルなビルドツールです。

RubyGems で配布されている Ruby 製のツールであるため，誰でも利用できる．また Rake の特徴として [6]

- Ruby で記述できる
- Rakefile というファイルに一連の処理を定義 (このとき処理させる振る舞いを「タスク」と呼ぶ) し，実行させることが可能
- 言語内 DSL を採用 (DSL とは特定の領域で特化して設計された言語)

といった特徴によって、Rake は強力なツールであるとされている．本研究では Rakefile にて処理を一括で管理することでユーザーフレンドリーなプログラムを目指した．

2.3.2 install

Rake を実行する際に Ruby がインストールされている必要がある．

```
$ gem install rake
```

と shell 上で入力すれば取得できる．

2.3.3 コマンド一覧

Rake で使用したコマンドは大きく分けて以下の通りである．

```
$ rake <タスク名> # 指定のタスクを実行
$ rake -help # 使い方を表示，これら以外にも複数コマンドが存在する
$ rake -T # 定義されているタスク一覧を表示
```

2.3.4 Rakefile の記述例

Rake は rake コマンドに引数として，実行したいタスクを渡すことでタスクを実行する．Rakefile に記述してあるタスクの記述方法は以下の通りである．

```
desc <タスクの説明，"rake -T"で説明が表示される>
task :<タスク名> do
  <処理する内容>
end
```

2.3.5 使用例

実際にプログラムを記述した例が以下の通りである.

```
1  desc "puts 'Hello, World!'"
2  task :hello do
3    puts "Hello, World!"
4  end
```

そして実際に使用した例が以下の通りである.

```
$ rake -T
rake hello  # puts 'Hello, World
$ rake hello
Hello, World!
```

上記のような形で使用する.

2.4 GitHub

管理対象であるデータを収集するにあたって，一箇所にデータを集めることが有効であると考え，Github は，Git の仕組みを採用したプログラムの元となるソースコードまたは画像データ等をインターネット上で管理・共有することが可能な開発プラットフォームである．本研究では，同研究室生のデータを収集，また Jekyll のプログラムコードを管理するため GitHub を用いて開発を進めた.

2.5 GitHub Pages

ウェブページを公開するにはサーバーが必要である．Github Pages[7] は，GitHub が提供するホスティングサービスで, GitHub のリポジトリを用いることで静的なウェブページをインターネット上に公開することが可能である [8]．ウェブページのアップロードにおいて Git コマンドを用いれば別途で FTP クライアントソフト (FTP は File Transfer Protocol の略称でファイルの送受信を行うための決まり) などを必要とすることがなく, 変更のあったファイルだけを更新することで閲覧可能である．また独自のドメインを設定することも可能である．これを用いることで Jekyll にてデータを閲覧できるようにした.

2.6 Jekyll

2.6.1 about

Jekyll は Ruby で書かれた静的サイトジェネレータで RubyGems で配布されているツールである．Jekyll を実行させるためには,

- コマンドラインツール
- Ruby

が必要である．Web サイト内で使い回すレイアウトといった繰り返し使う共通要素と，プレーンテキストを記述しておくことでコマンドを実行することで，それらを組み合わせて Web サイトを構成する HTML ファイルに変換してくれるものである [10]．HTML がおけるサーバーがあれば公開できるため，Github と連携することで GitHub Pages 経由で公開できる．

2.6.2 構造

本研究で作成した各プログラムによって生成される.md ファイルは Jekyll のディレクトリ直下に生成されるように設計した．Jekyll のディレクトリ構成は以下の通りで，ディレクトリ直下に置いてあるもので，それより下の階層のものは多量のため省略してある．

```
Jekyll
├── 404.html
├── Gemfile
├── Gemfile.lock
├── _config.yml
├── _data/
├── _includes/
├── _layouts/
├── _posts/
├── _sass/
├── _site/
├── about.md
├── assets/
├── blog.md
├── csv/
├── img/
├── index.md
├── life.md
├── text/
└── text.md
```

ディレクトリ内にある各ファイルやフォルダに関する役割を以下に記す．

_config.yml　サイトの設定を記述するものである．サイトのタイトルやサイトの説明文，Github のアカウント名等を反映させることができる．

_data　Jekyll のサイトのカテゴリーをこのフォルダ内にある navigation.yml を編集することでどの.html に遷移させるか設定することができる．テキストをリスト化してあるサイトを「text」, 生活習慣を表や図にしたものを表示させてあるサイトを「life」といった様にカテゴリーを設定できる．

_includes　ヘッダーやフッターといった各ページに共通して登場する HTML を保存する．

_layouts　各ページで共通して使う HTML の構造化したレイアウトを保存する．

_post　ブログのような記事を Jekyll では Post と呼び，このフォルダ内に.md ファイルで記述したものを保存することで公開することができる．その際のファイル名に，

`[4 桁の西暦]-[2 桁の月]-[2 桁の日]-[ページのタイトル].md`
`例:2019-07-03-about-install-Jekyll.md`

といった形で保存しておく必要がある．

_sass　Sass[11][12] とは，Syntactically Awesome StyleSheet の略称で，CSS に対して機能を拡張した言語である．そのため，ウェブページのデザインやレイアウトについての書かれたものが保存されてある．

_site　この中に入れられているファイルはディレクトリ内にあるものをビルド時に HTML に変換したものが置かれてある．そのため，Jekyll の直下においてある.md ファイルやサイト表示に使うイメージファイルといったほとんどのものがコピーされ，ここに置かれてある．

assets　「_sass」に保存されてあるものを使わず，ウェブ上でダウンロードしてきた.css ファイルを

`Jekyll/assets/css/*.css`

といった形でディレクトリ内に置く．CSS とは Cascading Style Sheets の略で，ウェブページのデザインやレイアウトなどの見た目を指定するための言語である．HTML と組み合わせて使用される [13].

img,csv,text フォルダ　バッジをテーブル化する際やテキストをリスト化するための元データ，サイト内で使われてある画像ファイルを保存してあるフォルダである．Jekyll でウェブページを作成するために必須ではないが，本研究で使用されるデータを格納するため，自ら作成したフォルダである．

.md，.html ファイル　.md ファイルは各カテゴリーを表示させる内容を Markdown 形式で記述したものである．これらを HTML 形式に変換することでウェブ上で閲覧することが可能である．ディレクトリ内にある index.md,text.md,life.md は，バッジテーブルやテキストのリスト，生活記録をチャート化したものを表示させるためにプログラムで作成されるように設計した．.html ファイルは.md ファイルを変換してできたものである．

2.6.3 install

Jekyll のインストール方法は Jekyll 公式サイトの方法を引用する．OS ごとによってインストールが違うため，ここでは macOS での方法を記述する．

Jekyll のインストール Bundler と Jekyll をインストールするだけです。

ローカルインストール

```
$ gem install --user-install bundler jekyll
```

それから Ruby のバージョンを取得します。

```
$ ruby -v
ruby 2.6.3p62 (2019-04-16 revision 67580)
```

それから、パスファイルに以下を追加します。X.X には、あなたの Ruby バージョンの最初の 2 桁を入れてください。

```
$ export PATH=$HOME/.gem/ruby/X.X.0/bin:$PATH
```

gem パスがホームディレクトリを指していることを確認するには、次のコマンドを実行します。

```
$ gem env
```

GEM PATHS:がホームディレクトリを指していることを確認してください。最初の 2 桁が異なる Ruby バージョンに更新する度に、パスもあわせて更新する必要があります。

グローバルインストールファイルのパーミッションの問題を解決し、sudo を使用するために Ruby Gems をグローバル環境にインストールすることを、強く推奨します。

Mojave(10.14) の場合 Mojave の SIP プロテクションのため、以下を実行しなければなりません。

```
$ sudo gem install bundler
$ sudo gem install -n /usr/local/bin/ jekyll
```

Mojave 以前（<10.14）次を実行するだけです。

```
$ sudo gem install bundler jekyll
```

2.7 draw.io

卒業論文の作成にあたって，システムの流れが分かりやすいようフローチャートとクラス図を作成した．その際に draw.io を使用した．draw.io[14] はフローチャートやクラス図の他にオフィスのレイアウトなどの図を作成することができるツールである．Web 上で GUI で作成し，png や jpeg などで出力できる．Excel や PowerPoint ではデフォルトで用意されていない素材が豊富にあるため，質の高い図を作成することができる．保存先に GitHub といったクラウドサービスにも対応しており，日本語でも使用できるため便利である．

第3章　各システムの実装

3.1　バッジ集めについて

3.1.1　バッジの採用理由

学生がどこまで学習しているか他の学生や教示者はすぐに把握することができない．そこで即座に確認することができる手段を考えた．そこでデジタルバッジ (以下，バッジと称す) に着目した．バッジは GitHub の README やマイクロソフトや IBM でスキルの証明に使用されている．本学の教授紹介でも SDGs について示す手段としてバッジが使用されるといった，多くのところでバッジを使用するケースが増えている．こういったバッジを利用される背景として，自分の能力を簡単かつ信頼性の高い方法でオンラインで共有することが可能だからである．先に述べた通り，Github では README.md でのバッジ表示が流行している. 理由としては,

1. リポジトリ管理が楽,
2. バッジにリンク付け，クリックするとページに飛ぶことができる,
3. バッジを保有することで自分の能力を簡単かつ信頼性の高い方法でオンラインで共有できる,

などがある. これを継続的に利用できないかと考えたところバッジ集め≒スタンプ集めに至った．多くの子供達は，朝のラジオ体操をはじめとしたスタンプ集めを体験している. スタンプ集めを行うことで,

1. 回遊性の向上
2. 滞在時間の向上
3. 副次的効果，満足度

といったメリットがある [15]．そのため, 人はスタンプ集めに夢中になる．またスタンプカードシステムよりスタンプを集め，学生の自主学習を促す研究が行われ，学習した内容をまとめさせることで自主学習の向上につながっている [16]. 現代ではゲーム内でもバッジを集めることで強さを証明する手段の一つとして考えられ，達成感が満たされていく．ゲームと同じような環境を作ることで親近感を沸かせることで，学習への意欲に繋げれないかと考えた．そこでスタンプ集めとバッジ表示 2 つの優位性を利用し,「バッジを集めることでゼミの出席を管理する」システムの開発を行うことを目標とした．開発するシステムで学生達の学習状況の管理を図った．

3.1.2 設計

バッジのテーブル化を行うプログラムを make_badge と命名し，CUI のように shell 上で make_badge と入力することで.csv からデータを抽出して.md を自動作成するプログラムの作成を行った．元データの形式は date, tutor, learner, attend, title とした．元データとなる.csv ファイルは図 3.1となる．

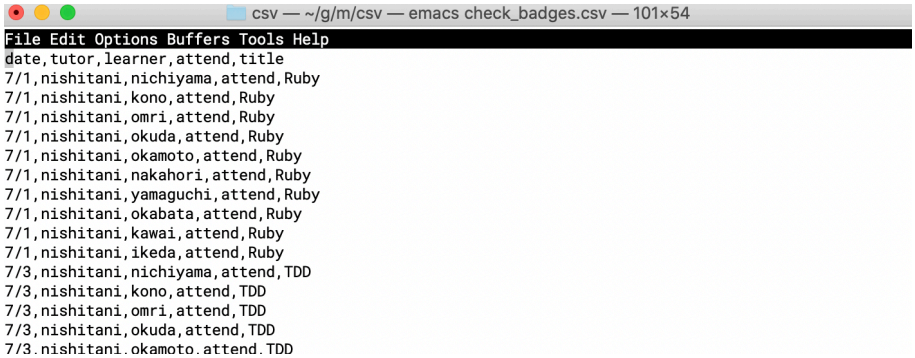

図 3.1: 元データの入力例.

バッジの作成に関しては，Shields.io を利用した．Shields.io はビルド結果やダウンロード回数などをバッジ表示できるサイトで，他のバッジ生成サイトに比べて種類が多い．また独自のバッジを作成することができる [16][17]． それにより，LABEL の部分に日付，MESSAGE の部分に学習したもののタイトル，COLOR を tutor によって色の値を設定した．図 3.2の URL パターンで値を設定すると作成できる．

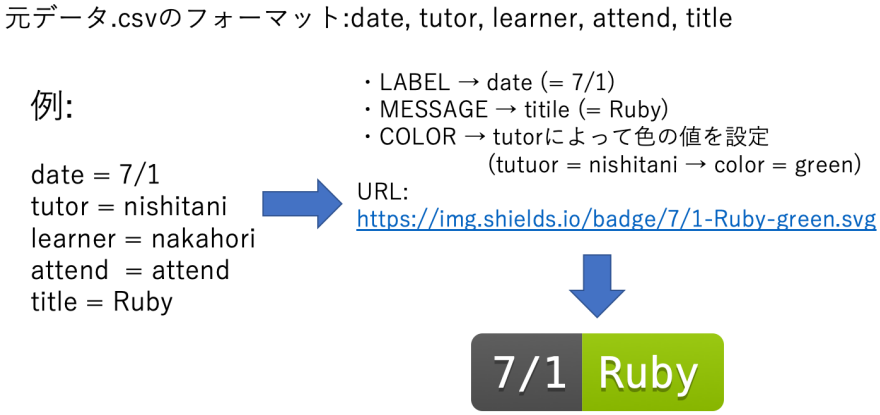

図 3.2: バッジの項目と生成されるバッジの例.

3.1.3 結果

Jekyll でのディレクトリに直接 index.md として出力されるようにした．生成されたバッジテーブルが図 3.3である．バッジの色を tutor ごとに色分けし，テーブル上に誰が教

えたか分かるように"tutor's color"として表示するように工夫した．こうすることで瞬時に誰が何を教えたか判断することができる．

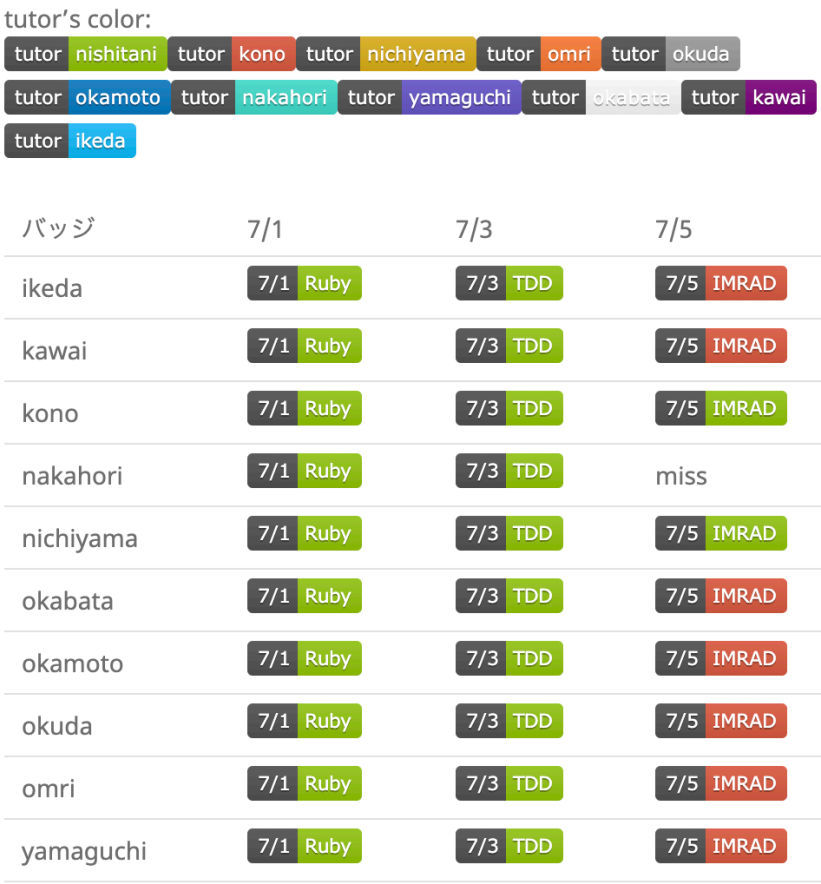

図 3.3: Jekyll での表示したバッジテーブル．

3.2 テキストのリスト化

3.2.1 リスト化にするメリット

西谷が開講するマルチスケールシミュレーション特論で使われるテキストがあり，これは同研究室生も学習している．Github にテキストとして html と org の二種類で用意されており，クローン (リモートリポジトリをローカルリポジトリにコピー) することで利用している．org より html の方がフォーマットが見やすく，リンク遷移も可能であるため，多くの学生がこちらを利用している．だがテキストのディレクトリが階層構造になっているため，html を開くのが手間である．そこで見やすくなるようテキストごとにリスト化した．これを Jekyll にて表示することでバッジテーブルで学習した内容を確認し，自分が次に学習する内容が何か分かりやすくした．

3.2.2 mk_list の実装

「mk_list」によってテキストを Jekyll にリストで表示する．この時，CoC(Convention over Configuration:設定より規約) という標語にならい，ディレクトリ名を使って階層管理を行った．テキストごとにソートをかけ，指定したディレクトリ内 (それより下にあるサブディレクトリ内) にある.html ファイルをリスト化し，.md ファイルに出力する．図 3.4は Jekyll にてリスト化したものを表示したものである．

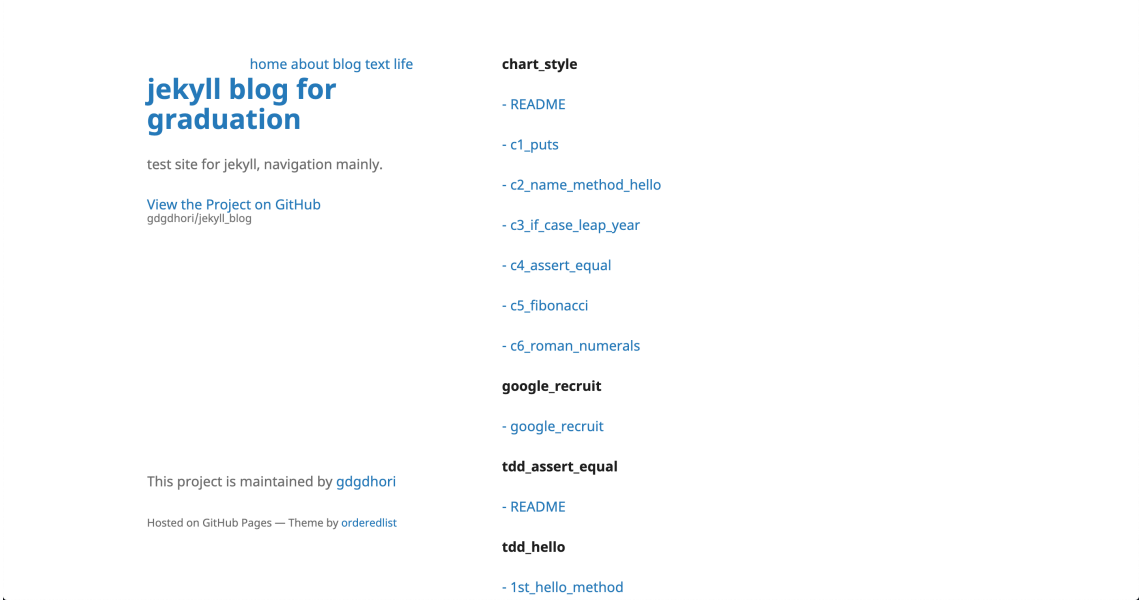

図 3.4: テキストごとにリスト化し Jekyll で表示した図.

各テキストのタイトルをクリックすることでローカルディレクトリに保存されてある html にリンクされるようにしてある．こうすることで煩わしいディレクトリ操作をすることがなく，かつローカルでも学習することが可能である．またバッジテーブルを参照することで自分がどこまで学習したか確認できる．

3.3 カレンダーアプリによる学習管理

3.3.1 設計

西谷が開講する大学院の講義にて同研究室生は参加することとなっていた．だが研究室生の参加率は悪かった．そこで学習管理の根本的な原因を調べるため，学生の生活記録を録り，自分がどのような生活を送っているか確認・比較することにした．そこで，まず日常記録を残していく必要がある．カレンダーアプリより日常の記録をとっていく．日常の記録のとり方については，後述の「データの入力について」にて詳述する．インターネットを通じてスケジュール情報をやり取りするための「iCalendar」と呼ばれるデータ形式で用いられる拡張子である.ics ファイルで出力し，.csv ファイルにて各項目ごとの時間の

合計時間をまとめる [18]．そして，この.csv ファイルからデータを読み取り，.png イメージにてパイチャートと積み上げグラフにて，生活記録の比較を行えるようにした．

3.3.2　データの入力について

カレンダーアプリより日常の記録を集計し，図や表にプロットする．そのため，パソコンやスマートフォンにインストールされているカレンダーアプリより日常の記録を残していく．各自,「ショッピング」「飲み会」といった好きなようにタイトルをつけて記録をとったが，種類が多く分かりづらかったため，研究者の視点でカテゴリーに分けるようにした．その結果,

- Work(勉強や就活等)
- Game(ゲームや動画鑑賞，友人と遊ぶ等)
- Sleep(睡眠や昼寝等)
- Eat(朝食，昼食，夕食等)
- Transfer(運転，交通公共機関等)
- Other(アルバイト，病院等上記の項目に含まれないもの)

の 6 つに分類した．したがって上記の項目に従って分単位で記録を残していった．カレンダーに記録を付けていくとプライベートと混ざって分かりにくくなるかと心配になるが，カレンダーアプリは新規カレンダーとしてプライベートと研究用とで記録するカレンダー先を区別することができるため，研究用として記録を付けていった．実際に記録をつけていった例が図 3.5,3.6である．

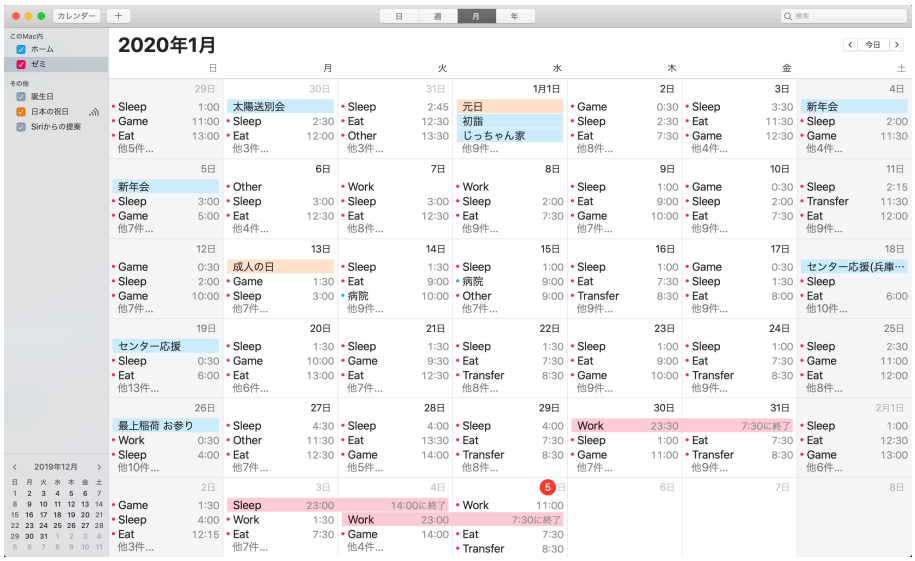

図 3.5: 記録をつけていったカレンダー．

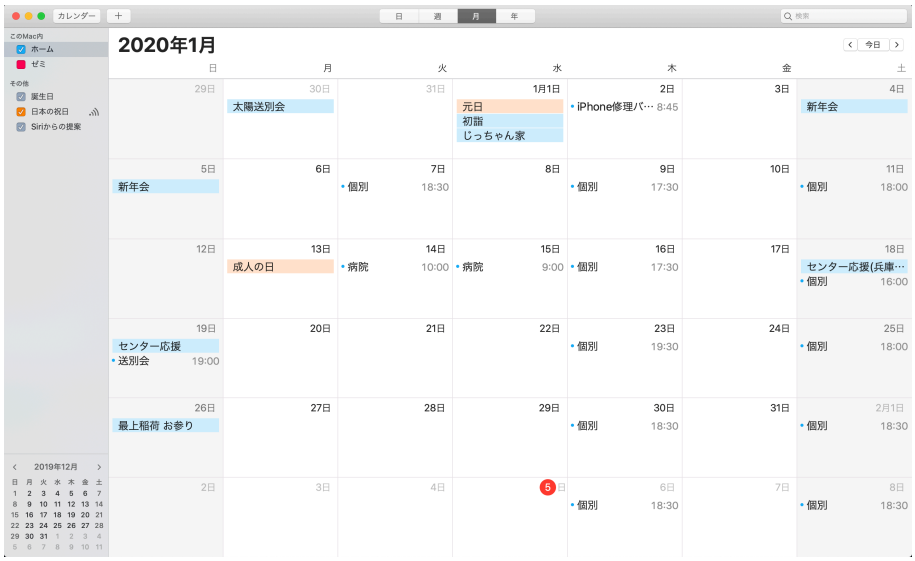

図 3.6: 研究用記録を非表示にしたカレンダー.

3.3.3 Liferecord の実装

Rakefile のコマンドについて 作成した Rakefile のコマンド一覧は以下のようになる.

```
$ rake -T
rake csv2png_p        # convert csv to piechart   of personal
(ex: rake csv2png_p [#filename])
rake gather_ics_all   # gather all .ics (copy all .ics  from
other locations to ics folder)
rake gather_ics_p     # gather .ics (ex: rake gather_p [#filename])
rake ics2csv_p        # convert ics to csv   of personal(ex: rake
ics2csv_p [#filename])
rake ics2csv_p_t      # convert ics to csv   of personal in term
(ex: rake ics2csv_p [#filename])
rake ics2csv_total    # convert ics to csv data of one month life
rake ics2png_all      # convert ics to png in all
rake ics2png_p        # convert ics to png (ex: rake ics2png [#filename])
rake stacked_graph    # plot stacked graph
rake view             # view Jekyll
```

コマンドの末尾ついているアルファベットの意味を補足説明をする.

```
$ rake [:command]_p [:args]
```

"_p"と記述のあるコマンドについては 1 つ (1 学生) のファイルを操作したものとなっている. 通常の Rakefile とは引数の受け取り方が違い操作しやすいように引数のように扱えるようにした. そのため, 一番最後に入力した引数 (ARGV.last) が扱われるように設計してある.

```
$ rake [:command]_all
```

"_all"と記述のあるコマンドについては全てのファイル (全学生) を操作するようにしたものとなっている．そのため引数は必要としない．
　後述より各コマンドの説明である．

gather ics　.ics ファイルを集める二種類のコマンドを用意した．

```
rake gather_ics_all  # gather all .ics (copy all .ics  from
other locations to ics folder)
rake gather_ics_p    # gather .ics (ex: rake gather_p [#filename])
```

　"gather_ics_all"は"gather_ics_p"の対象を全てにしたもので，指定したディレクトリ内 (それより下にあるサブディレクトリ内) にある全ての.ics ファイルを ics フォルダにコピーする．2 つのコマンドの例は以下の通りである．入力例は以下のようになる．

```
$ rake gather_ics_all
copy nakahori.ics in ics
copy omri.ics in ics
copy ikeda.ics in ics
copy kawai.ics in ics
copy okabata.ics in ics
```

　"gather_ics_p"は指定したディレクトリ内 (それより下にあるサブディレクトリ内) にある個人の.ics ファイルを ics フォルダにコピーする．入力例は以下のようになる．

```
$ rake gather_ics_p nakahori
gather_ics_p
copy nakahori.ics in ics
```

convert ics to csv　.ics ファイルから.csv に変換する三種類のコマンドを用意した．

```
rake ics2csv_p       # convert ics to csv  of personal(ex: rake
ics2csv_p [#filename])
rake ics2csv_p_t     # convert ics to csv  of personal in term
(ex: rake ics2csv_p [#filename])
rake ics2csv_total   # convert ics to csv data of one month life
```

　"rake ics2csv_p"は個人の.ics ファイルのデータから前日から 1 週間の生活記録の各項目ごとの合計時間を秒単位にて集計したデータを.csv ファイルに入力する．入力例は以下のようになる．

```
$ rake ics2csv_p nakahori
convert nakahori.ics
create nakahori.csv
```

```
$ cat csv/nakahori.csv
2020-01-19,2020-01-26
Sleep,212400
Eat,70200
Other,102600
Transfer,70200
Game,89100
Work,155700
```

"rake ics2csv_p_t"は個人の.ics ファイルのデータから指定した期間の生活記録の各項目ごとの合計時間を秒単位にて集計したデータを.csv ファイルに入力する．入力例は以下のようになる．

```
$ rake ics2csv_p_t nakahori
convert nakahori
since when?(ex:20-01-01)
20-01-14
until when?(ex:20-12-31)
20-01-24
create nakahori2.csv
$ cat csv/nakahori2.csv
2020-01-14,2020-01-24
Sleep,255600
Other,154800
Eat,100800
Work,103500
Game,191700
Transfer,57600
```

"rake ics2csv_total"は個人の.ics ファイルのデータから一週間単位の生活記録を 4 週間分 (1ヶ月分) の各項目ごとに合計時間を秒単位にて集計したデータを.csv ファイルに入力する．入力例は以下のようになる．

```
$ rake ics2csv_total nakahori
convert nakahori.ics
create nakahori.csv
$ cat csv/nakahori_total.csv
1/27~2/2,1/20~1/26,1/13~1/19,1/6~1/12
Eat,21600,52200,75600,55800
Game,40500,78300,171000,113400
Other,30600,75600,115200,86400
Sleep,108000,192600,171000,198900
Transfer,21600,61200,36000,48600
Work,137700,144900,36000,101700
```

convert csv to png　.csv ファイルから.png に変換する一種類のコマンドを用意した．

```
rake csv2png_p        # convert csv to piechart   of personal(ex: rake
csv2png_p [#filename])
```

"rake csv2png_p"は各生徒の.csv ファイルを.png イメージのパイチャートで出力させる．入力例は以下のようになる．

```
$ rake csv2png_p nakahori
convert nakahori
create nakahori.png
```

生成された.png は図 3.7のようになる．

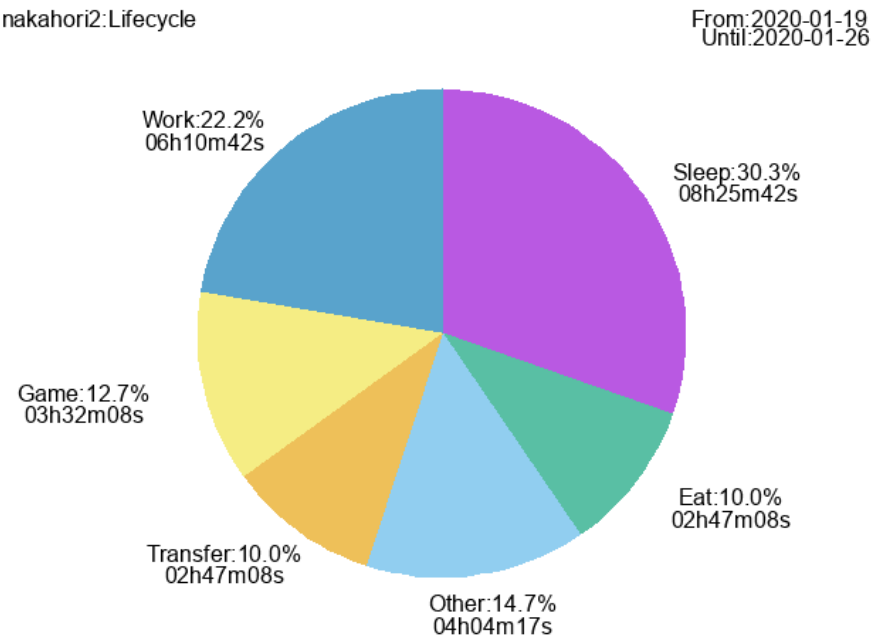

図 3.7: rake csv2png_p にて生成された.png ファイル．

.csv ファイルにデータが入力されていない場合，途中で処理が終了する．

```
$ rake csv2png_p omri
convert omri.csv
no data in the csv file   of omri.
```

convert ics to png　.ics から.csv へ変換し，.csv から.png へ変換しパイチャートを出力していると手間に思う方がいると思う．そこで一度に.ics から.png へ変換するコマンドを二種類用意した．

```
rake ics2png          # convert ics to png (ex: rake ics2png [#filename])
rake ics2png_all      # convert ics to png in all
```

"rake ics2png_p"が各学生の.ics ファイルを.png イメージのパイチャートで出力させたもので引数 (ファイル名) を必要とする，入力例は以下のようになる．

```
$ rake ics2png_p nakahori
copy nakahori.ics in ics
convert nakahori.ics
create nakahori.png
```

"rake ics2png_all"は全学生の.ics ファイルを.png イメージのパイチャートで出力させるもので全て学生を対象とするので引数を必要としないある．入力例は以下のようになる．

```
$ rake ics2png_all
cope nakahori.ics in ics
convert nakahori.ics
create nakahori.png
cope ikeda.ics in ics
convert ikeda.ics
no data in the csv file   of ikeda.
```

stacked_graph　.ics ファイルから.png イメージにて積み上げグラフを出力させるコマンドは以下の通りである．

```
rake stacked_graph   # plot stacked graph
```

"rake stacked_graph"は.ics ファイルから 4 週間分の生活記録を集計し，.csv ファイルに出力し，そして.png イメージにて積み上げグラフにしてプロットする．プロットしたものが図 3.8となる．

view Jekyll　Jekyll にて出力した.png イメージを html にて表示させるコマンドは以下の通りである．

```
rake view            # view Jekyll
```

作成した.png イメージを Jekyll を通して閲覧する．Jekyll はページを拡張子.md にして Markdown で書くこともでき, ビルド時に HTML に変換し，閲覧することができる．その際に，作成した.png イメージを Jekyll のディレクトリ内の指定した画像フォルダにコピーを行い，.md ファイルに入力する．そして編集したデータを Github にアップロードを行う．こうすることでデータが反映され，インターネット上で閲覧することができる．実際に表示されたページが図 3.9である．

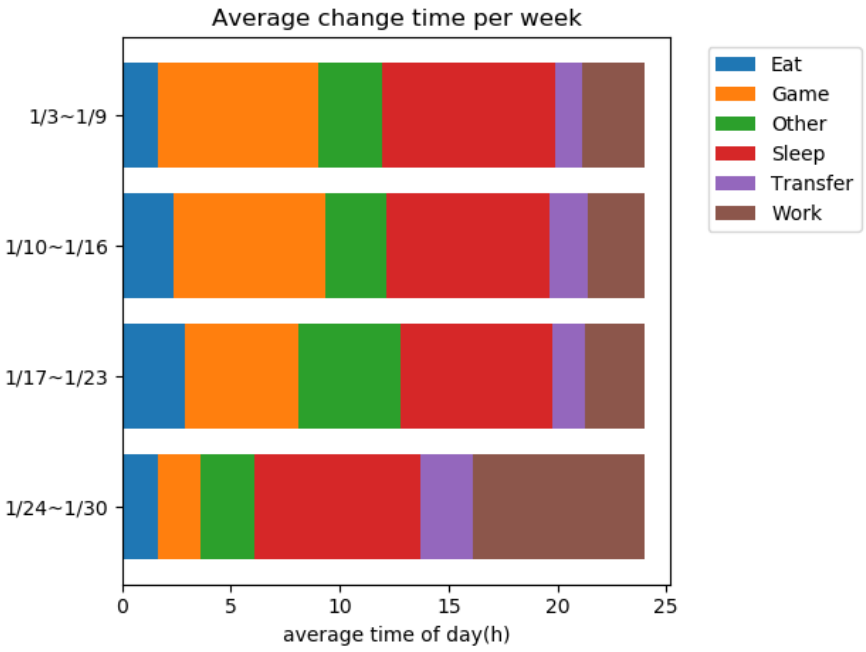

図 3.8: .ics から作成した積み上げグラフ.

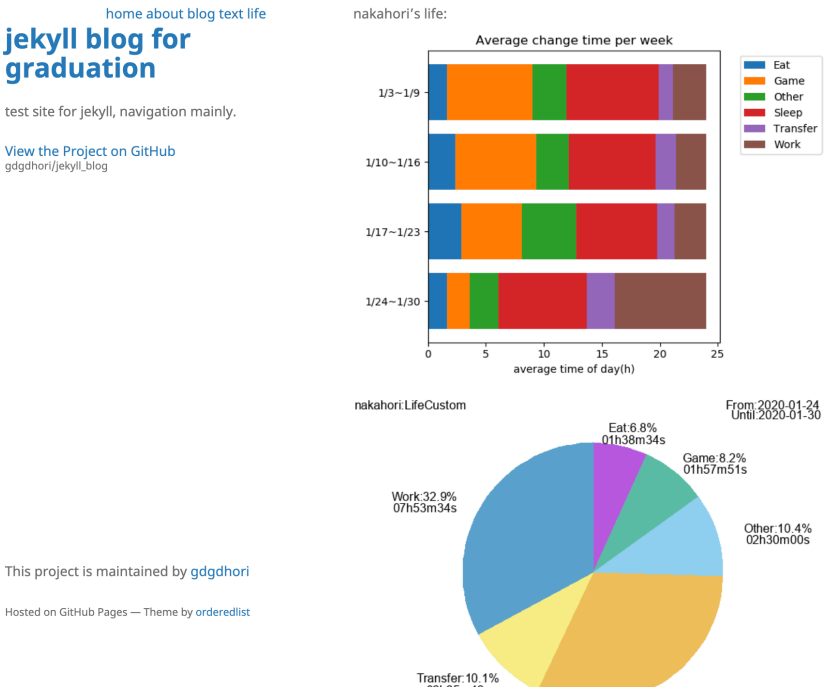

図 3.9: Jekyll にて表示されたグラフ.

第4章　考察

4.1　評価

バッジテーブル表示により，誰が誰に何を教えたかが分かるようになった．テーブル化することにより誰が何を学習したのか一目で分かるようになり，そのテーブルを見ることで，誰に何を聞けばいいか分かるようになった．もし大学を欠席することがあった場合，バッジが付与されている生徒に聞くことで，欠席した分を補うことができる．

また，前回自分がどこまでやったのか確認し，テキストのリストから次に何をやればいいのか判断でき，円滑に遷移できた．

さらにカレンダーアプリを用いた生活の記録を図や表にして日常の記録を残していくことにより，自らの生活を振り返ることができた．学習以外の項目も図や表にして見ることができるため，他の項目と比較して見ることができた．これにより自分が何の項目の時間が減り，何の時間が増えたか比較することで，今後の生活改善に役立てることができる．可視化することによって教示者から学習時間を指摘されることで学習時間の改善につながった．この2つを利用することで，教示者が課題を与えられ，達成できれば次の内容に進み，達成できなかったり，休んでた場合，バッジを所持している者に指導してもらったり，どれだけ学習して課題が達成できた，達成できなかったと教示者が他の受講者の学習量を把握することができる．

4.2　今後の課題

4.2.1　プログラムのテストとリンクしたバッジテーブル

現在バッジテーブルを作成する際に元データとなる.md ファイルは手入力での操作となる．毎回操作となると煩わしく効率の良い学習にはつながらない．また学習した際に作成したプログラムのテストコードがないため，学習者が書いたコードが本当に正しい振る舞いをしたのか分からない．そこで作成したプログラムをテストできるようにし，テストが完了すれば自動的に.md ファイルに追記されるように行う．これにより．自分の作成したプログラムにリンク付ければ見直すのが簡単になり，自分の学習した範囲を見直すのが容易になる．またバッジに次にやる内容のテキストをリンクづけしていれば，簡単に次にやる内容へと遷移することができ，円滑に学習に移ることができ，操作の幅が広がるのではないかと考えた．

4.2.2 カレンダーアプリへの入力の仕方

カレンダーアプリに項目ごとに毎回入力を行うが手間である．手間になるとカレンダーアプリへの入力を怠り，あとでまとめて記録をつけることになる．そうなると正確な記録をつけることが困難になる．そのため，毎日コツコツつけていくための工夫が必要となる．端末のスクリーンタイムの時間を取得して，カレンダーに反映できれば，携帯やパソコンを操作していた時間は少なくとも自動で入力ができるのではないかと考えた．

4.2.3 グラフの表示方法の多様化

スマートフォンで気軽に生活の記録をとれるアプリケーションとして,「ライフログ [19]」や「aTimeLogger[20]」といった様々なものがある．これらの他のアプリケーションと比較すると，表示される図や表の種類が圧倒的に少ない．各項目ごとにグラフがあったり，積み上げグラフと折れ線グラフが同時にプロットされてたり，他にストップウォッチの機能があったりと機能に富んでいた．そのため，各項目ごとの変化が分かるようグラフの種類を増やし，時間を測り，記録できるようなシステムを作ることが今後の課題である．

第5章　総括

本研究では，バッジ集めまたテキストのリスト化，カレンダーアプリからの日常生活の記録による学習管理を行った．開発したシステムをまとめると，

- バッジテーブルによる徒弟制による学習管理,
- 複数台の PC からのデータ収集,
- データを元にしたチャートの生成,
- Rakefile の作成，コマンドの実装,
- Jekyll にて作成したテーブルやチャートの表示,

である．上記により学習管理が容易になった．

バッジテーブルを参照することで誰が誰に教えてもらい，なにを学習したのかが分かり，バッジを参照することで次に学習する内容を把握することができる．もしゼミを欠席したり，与えられた内容をこなすことができなかった場合，カレンダーアプリへ生活記録を残すことで何故欠席したのか，またどれだけ学習していたのかを把握することができる．

今後の展望としては，プログラムのテストとリンクしたバッジテーブルを作成することにより，作成したプログラムを即座に見直せるようにし，学習効率の向上につなげる．また，カレンダーアプリへの入力が手間なため，スクリーンタイムでのデータをカレンダーアプリへ反映させることで日常記録の入力の手間を減らすことを可能にし，さらに表示させる図や表の数を増やし，細かなところまで変化が分かるように機能を増築したシステムを作成することが改善に繋がるのではないかと考える．

謝辞

本研究を行うにあたり，終始多大なる御指導，御鞭撻をいただいた西谷滋人教授に対し，深く御礼申し上げます．また，本研究の進行に伴い，様々な助力，知識の供給を頂きました西谷研究室の同輩，先輩方に心から感謝の意を示します．本当にありがとうございました．

参考文献

[1] "マイクロソフト試験と認定資格バッジ",Microsoft,https://www.microsoft.com/ja-jp/learning/badges.aspx ,(2020/02/04, accessed).

[2] "Master the Mainframe バッジ",IBM,https://www.ibm.com/jp-ja/it-infrastructure/z/learn/master-the-mainframe-badges, (2020/02/04, accessed).

[3] 西城 卓也 (2012), "正統的周辺参加論と認知的徒弟制", 医学教育, 43 巻 4 号,p.292-293, 日本医学教育学会,https://www.jstage.jst.go.jp/article/mededjapan/43/4/43_292/_pdf, (2020/02/04, accessed).

[4] "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", コトバンク,https://kotobank.jp/word/ユーザーフレンドリー-9485, (2020/02/04, accessed).

[5] library rake,https://docs.ruby-lang.org/ja/latest/library/rake.html, (2020/02/04, accessed).

[6] "Rails の rakeってなんぞ？",SuguruOoki,https://qiita.com/SuguruOoki/items/e736b15bbb80eacf66d7, (2020/02/04, accessed).

[7] "無料で使える！GitHub Pages を使って Web ページを公開する方法",TECHACADEMY,https://techacademy.jp/magazine/6445, (2020/02/04, accessed).

[8] Github Pages, https://pages.github.com/,(2020/02/04, accessed).

[9] RubyGems, https://rubygems.org/,(2020/02/04, accessed).

[10] Jekyll, https://jekyllrb.com/,(2020/02/04, accessed).

[11] Sass, https://sass-lang.com/,(2020/02/04, accessed).

[12] "CSS の基本",HTML クイックリファレンス,http://www.htmq.com/csskihon/001.shtml,(2020/02/04, accessed).

[13] Assets,Jekyll, https://jekyllrb.com/docs/assets/,(2020/02/04, accessed).

[14] draw.io,https://www.draw.io/, (2020/02/05, accessed).

[15] "シヤチハタスタンプラリー", スタンプラリー研究所,https://www.shachihata.co.jp/stamprally_service/lab/5.html, (2020/02/04,accessed).

[16] HARDY Darrell(2015),"授業外で英語へ触れさせることを広げかつ自主的な学習を促し、課題を監督するためのスタンプカードシステムの導入", 観光学研究 = Journal of tourism studies, 14, p.75 - 97, 東洋大学国際地域学部,https://toyo.repo.nii.ac.jp/?action=repository_uri&item_id=7318&file_id=22&file_no=1, (2020/02/04,accessed).

[17] Shield.io, https://shields.io/,(2020/02/04, accessed).

[18] "Github の README.md をバッジでオシャレにできる Shields.io と dockeri.co",kakku22, https://kakakakakku.hatenablog.com/entry/2018/08/08/200903,(2020/02/04, accessed).

[19] ".ics",BINARY 拡張子辞典, https://www.sophia-it.com/extension/content/.ics, (2020/02/04, accessed).

[20] "ライフログ-食事-運動-睡眠サイクルを記録して分析する生活習慣管理アプリ",Picup Inc., https://apps.apple.com/jp/app/ライフログ-食事-運動-睡眠サイクルを記録して分析する生活習慣管理アプリ/id1090728028, (2020/02/07, accessed).

[21] "aTimeLogger-パーソナル時間記録",BGCI,https://apps.apple.com/jp/app/atimelogger-パーソナル時間記録/id576718804, (2020/02/07, accessed).