卒業論文

# hikiutils を用いた<br>卒業論文作成

関西学院大学　理工学部　情報科学科

1234 西谷滋人

2017 年 3 月

指導教員　　西谷　滋人　教授

# 目次

# 目次

## 1 ユーザメモと wiki を連携するシステムの開発

### 1.1 関西学院大学 理工学部 情報科学科 3550 西谷研究室 江本沙紀

近年，ナレッジマネジメントが企業経営の重要な要素と言われ，導入する企業が増えている．ナレッジマネジメントとは，個人の持つ知識や情報を組織全体で共有し，有効に活用することで業績を上げようという経営手法である．日本語では，「知識管理」などと訳され，「KM」と略されることもある．参考文献 1(`http://e-words.jp/w/`ナレッジマネジメント.html 1/27 アクセス)

ナレッジマネジメントでは，グループ開発において共有する知識は暗黙知と形式知に分けられる．暗黙知は主に口伝によって一対一でつたえられたり，あるいは体で覚えるというのが一般的である．しかし，定着するまでの間は一般的にメモという形で個人的な知識として扱われるのが普通
誰もが読める形で保管，

形式知と暗黙知

| 形式知 | 暗黙知 |
| --- | --- |
| 言葉や文字で表現できる<br>記録として残されている | 言葉や文字で表現できない<br>無意識や筋肉記憶の中に存在 |
| 多くの人に伝えるときに効果的 | 人と人のコミュニケーションによってのみ伝達可能 |
| (例)料理本，教科書 | (例)包丁の使い方，火加減 |

図 1　暗黙知と形式知

参考文献 2(ニック・ミルトン，「プロジェクト・ナレッジ・マネジメント」，(生産性出版，東京都渋谷区渋谷 3-1-1，2009 年)，p.4-5)

暗黙知の形式知化はいくつも行われている．google 検索でよく引っかかる Qiita.com や cheatgraphy.com などもそれらをまとめるサイトを提供している．また，デザインパターンも「プログラマが持っていた暗黙知に名前をつけることで形式知化した」と ruby 開発者のまつもとゆきひろも指摘している`{{fn,`**デザインパターン，コード解説**`}}`．ある意味，暗黙知の形式知化をいかに効率よく行うかは知識共有の最大の目的とも言える．

このような暗黙知と形式知を提供するフォーマットはそれぞれの特徴を引き出すためにそれぞれ異なったフォーマットで記述されている．

- 暗黙知は主にメモとして保存しやすいように単純なテキスト形式が取られる．
- web 発信においては，複雑な構文となる hyper text ではなく，手軽に web サイトを構築する wiki に対応した mark up 言語が取られる．また，
- 書籍としては，体裁，数式の綺麗さだけでなく，目次，索引，引用文献などの自動作成の観点から latex で書かれることが多い．

しかし，それぞれを別々に書いていては，一箇所に修正があるとすべてのフォーマットに対して行う必要が出てくる．これはプログラマの心得の核心をなす「DRY(Don't Repeat Yourself)」原則を破ることとなる{{fn('pragmatic programmer')}}．プログラマはこれらの変換を自動化するコンバータを作成して，一箇所の修正によって他のフォーマットでの修正に反映されるように，自動的に行うシステムを構築している．

本研究においては，図 2 に示した通り，西谷研で活用しているメモソフト my_help, wiki clone の hiki, および latex の間を自動変換するシステムの開発を目的としている．

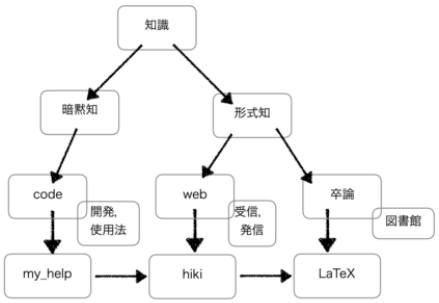


図 2　知識の分類と，それぞれに適合したシステムおよびフォーマット．

# 2　関連する先行研究

### 2.0.1 ユーザ独自の help を作成する gem

■gem とは　正式名称は RubyGems．Ruby 用のライブラリを使う時に必要となるソフトウェアのこと．パッケージ管理ツール gem があることで，Ruby 用ライブラリのインストール，アンインストール，バージョン管理などを簡単に行うことができる．プログラミング言語 Ruby のファイルに付属されていて，無料で利用することができる．

■gem の利点

- 標準化された構造があるので，初めてみた人でも分かるようになっている．
- gem があることで，簡単に Ruby 用ライブラリをインストールでき，初心者でもアプリ機能を装備できる．
- 誰でも作成，配布が可能である．

■my_help とは　西谷研究室で使用している gem.

```
Usage: my_help [options]
    -v, --version                    show program Version.
    -l, --list                       list specific helps
    -e, --edit NAME                  edit NAME help(eg test_help)
    -i, --init NAME                  initialize NAME help(eg test_help).
    -m, --make                       make executables for all helps.
    -c, --clean                      clean up exe dir.
        --install_local              install local after edit helps
        --delete NAME                delete NAME help
        --hiki                       my_help2hiki
```

上記のようなコマンドが用意されていて，ユーザ独自の help(メモ) を作成することができる．

■my_help を研究室内で利用する利点

- 研究室内でのメモの書き方が統一できる．
- どこにメモをしたか忘れることがない．
- 普段研究の為に使うターミナルから離れること無くメモを残すことができるので，書きたいときにすぐに書くことができる．考文献 (https://blog.codecamp.jp/rails-gem

1/27)

### 2.0.2 hiki とは

Ruby で書かれた高機能・高速 Wiki クローン．参考文献 3(http://hikiwiki.org/ja/ 1/27) CGI(Commond Gateway Interface) を利用して，Web サーバと連動して動く．参考文献 4(https://ja.wikipedia.org/wiki/Hiki 1/27) 西谷研究室では，hiki の形式を利用してサイトを作り，研究室内での情報共有や gem の使い方などを掲載して閲覧できるようにしている．また，卒業論文の作成にも hiki の形式で作成している．

### 2.0.3 TeX とは

「テック」または「テフ」と読む，組版ソフト．スタンフォード大学の Donald E.Knuth 教授によって作られた．他のソフトと組み合わせて，組版結果を画面や紙に出力したり，PDF 形式で出力したりする．

### 2.0.4 TeX の特徴

- フリーソフトなので無料で入手でき，自由に中身を調べたり改良したりできる．
- 商品利用が自由．
- Windows，Mac，Linux などの OS に関わらず同じ動作をする．
- TeX の文書はテキストファイルなので，普通のテキストエディタで読み書きでき，再利用やデータベース化が容易．
- 数式の組版に定評があり，数式をテキスト形式で表す標準となっている．

### 2.0.5 latex とは

「ラテック」または「ラテフ」と読む．DEC(現在の HP) のコンピュータ科学者 Leslie Lamport によって機能強化された TeX. もとの TeX と同様にフリーソフトとして配布されている．

参考文献 5「LATEX2e 美文書作成入門」，奥村晴彦/黒木祐介著，技術評論社，2016, p.1-3

### 2.0.6 gem 中の hikis から hiki への自動変換

ruby のライブラリーパッケージの標準となる gem の directory 構造に hikis という directory を作って文書作成している．hiki –initialize でこのなかの hikidoc ファイルをウェブ上の hiki と同期する機能を提供する．これによって，gem/hikis で作成した文書は，github あるいは rubygems.org を通じて共有可能となる．以下にこの同期をスムーズに行うための幾つかの convention を使用法とともにまとめている．

**■使用法，コマンド**

- hiki –initialize： 必要なファイル (Rakefile, ./.hikirc, hiki_help.yml) が copy される
- hiki_help.yml： 適宜 /.my_help に copy して hiki_help として利用 `my_help` 参照 (`MyHelp_install`)
- rake sync：hiki ディレクトリーと同期が取られる
- rake convert 20 TARGET.png：figs/TAERGET.png に 20
- hiki -u TARGET：ブラウザー表示される

**■同期に関する制約**

- hiki はフラットな directory 構造を取っている
- hiki の文書はスネーク表記 (例えば，latex2hiki_saki) で階層構造に似せている
- hiki の url の接頭語となる名前を basename の directory 名とする．
- directory 名が'hikis' である場合はその親 directory 名となる．
- /.hikirc の target directory を同期先の directory とする．
- /.hikirc がない場合は同期先の directory を聞く．
- それらは./.hikirc に保存される

**■テキスト**

- テキストの拡張子は'.hiki' としている
- hiki での url はテキスト前とディレクトリーから自動生成される
- 例えば，hiki2latex_saki/introduciton.hiki とすると hiki2latex_saki_introduciton と変換される
- attach_anchor では

```
'{{attach_anchor(test.png, hiki2latex_saki)}}'
```

と，directory 指定しなければならない.

### 2.0.7 hiki 形式で書かれた文章を latex 形式に自動変換する

**■コマンド**

```
Usage: hiki2latex [options] FILE
    -v, --version                    show program Version.
    -l, --level VALUE                set Level for output section.
    -p, --plain FILE                 make Plain document.
    -b, --bare FILE                  make Bare document.
        --head FILE                  put headers of maketitle file.
        --pre FILE                   put preamble file.
        --post FILE                  put post file.
        --listings                   use listings.sty for preformat with style
```

上記のようなコマンドがある.

**■使用例**

- hiki2latex -v

hiki2latex の version を表示する. 実行例

```
/Users/saki% hiki2latex -v
hiki2latex 0.9.12
```

- hiki2latex -p

hiki 形式の sample, hiki2latex_sample.hiki を例とする.

```
/Users/saki/my_help2hiki/my_help2hiki_saki% cat hiki2latex_sample.hiki
!title1
!!subtitle1.1
*list1
*list2
!!subtitle1.2
*list1
```

```
!title2
```

コマンドを実行すると以下のようになる.

```
/Users/saki/my_help2hiki/my_help2hiki_saki% hiki2latex -p hiki2latex_sample.hi
\documentclass[12pt,a4paper]{jsarticle}
\usepackage[dvipdfmx]{graphicx}
\begin{document}
\section{title1}
\subsection{subtitle1.1}\begin{itemize}
\item list1
\item list2
\end{itemize}
\subsection{subtitle1.2}\begin{itemize}
\item list1
\end{itemize}
\section{title2}
\end{document}
```

このコマンドにより，hiki 形式の雛形が生成されていることが分かる.

- hiki2latex -b

hiki2latex_sample.hiki を例とするコマンド実行例

```
/Users/saki/my_help2hiki/my_help2hiki_saki% hiki2latex -b hiki2latex_sample.hi
\section{title1}
\subsection{subtitle1.1}\begin{itemize}
\item list1
\item list2
\end{itemize}
\subsection{subtitle1.2}\begin{itemize}
\item list1
\end{itemize}
\section{title2}
```

上記の間に記述された部分のみを生成する．同じファイルの中に，複数の latex のファイルを含んでいるときに個別でファイル作成をするときに便利．

- hiki2latex –head： タイトル，著者を記述するときのフォーマットを挿入する．
- hiki2latex –pre： 文字の前の空白などを入れる．
- hiki2latex –post

上記 3 つは全てフォーマットを標準と違うものにするためのコマンド．

# 3　my_help，wiki，latex の自動変換システム

## 3.1 従来の memo,

| | 作成 | 閲覧 | 書式 | 目的 |
|---|---|---|---|---|
| my_help | ターミナル | ターミナル | yaml | ユーザメモ |
| wiki | mi | safari | hiki | web閲覧 |
| latex | TeXShop | pdf | TeX | 卒論，書類 |

図 3　memo，latex，hiki の比較

### 3.1.1 my_help

my_help は，メモを作るための gem．ユーザがターミナルを利用して作成し，閲覧する．メモの書式は yaml 形式で，拡張子は.yml を使用している．

### 3.1.2 wiki

mi という mac のテキストエディタを用いて作成する．作成したファイルは Mac OS のブラウザの safari で開くことができる．hiki 形式で記述し，web を通して誰でも見られるようにできる．

### 3.1.3 latex

TeX の書式で，TeX 編集用エディタの TeXShop によって作成する．pdf は一般的に使われている電子ファイルで，印刷して卒業論文のハードコピーとしても使われる．

このように my_help，wiki，latex は書式や作成，閲覧方法が全て違う．目的も異なるので，同じ内容のファイルをそれぞれの書式で書かなければならないことがある．

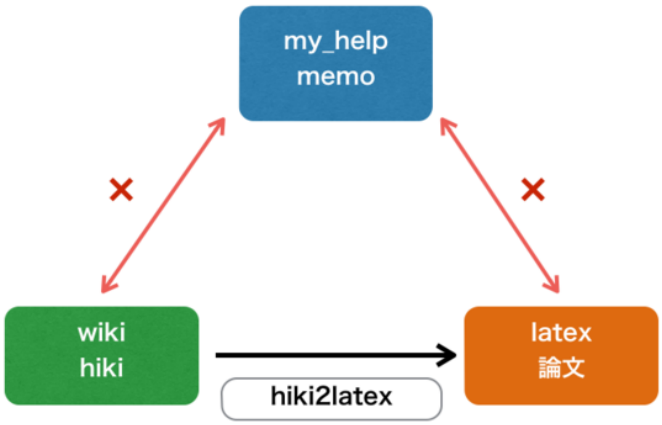


図 4　hiki2latex

hiki2latex の開発により上図のように wiki と latex の変換はできるようになったが，my_help と wiki，my_help と latex は関連させることができなかった．本研究により，my_help，wiki，latex を関連させるため，my_help から wiki へ自動変換を行うシステム my_help2hiki の開発を目指す．

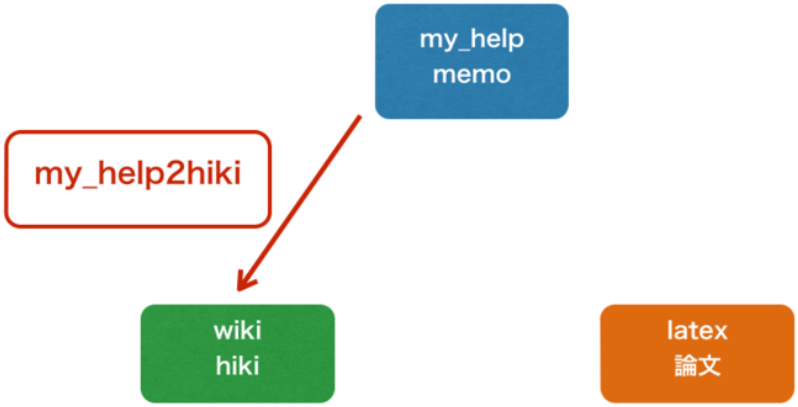


図 5　my_help2hiki

# 4　考察

#### 4.0.1　情報共有

メモを更新してから，my_help2hiki の２つのコマンド

- TARGET –push
- my_help –hiki

を共に行うと，自分の my_help にあるメモ，サーバのバックアップ，研究室内のメンバーが見ることのできる wiki の全てが同じ内容になる．研究室内全員が同じ情報を得ることで，情報を共有するという目標が達成できる．

#### 4.0.2　memo，hiki，lat

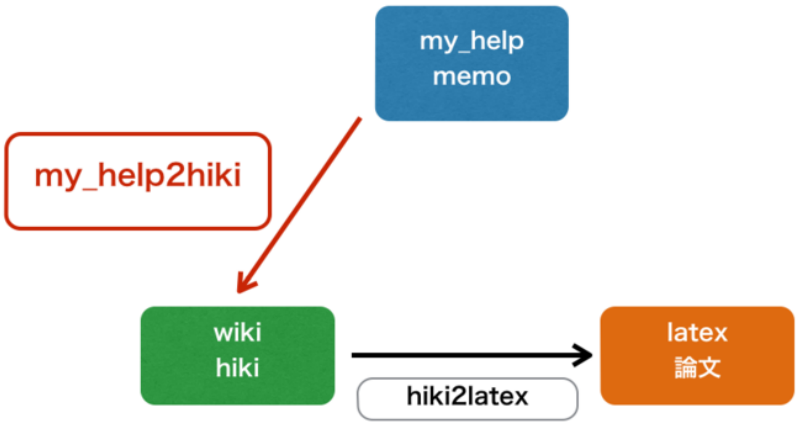


図 6　my_help2hiki による memo，hiki，latex の関係

上記の図のように my_help2hiki を利用することで，memo から wiki を自動変換が可能になり，my_help と hiki の連携ができるようになった．さらに hiki2latex を利用すれば wiki から論文の形の latex へと変換することもできるので，研究を進めている際に memo を作っていれば間接的に，卒業論文へ繋げることができる．

#### 4.0.3 暗黙知の形式知化

また，メモをすべて my_help で作成しておけば，忘れてしまったときに my_help のみを確認すればよく，管理も楽になる．gem の開発中や課題の途中にもすぐメモを残すことができるので，それぞれのユーザが得た暗黙知もすぐに形式知化することが可能になることが期待される．

#### 4.0.4 書き方が細かく定められている

誤って一つでも消してしまうとエラーがでて動作しなくなってしまう．emacs_help を例にする．emacs_help の内容は下のようになっている．

```
/Users/saki/.my_help% cat emacs_help.yml
---
:head:
- "emacs のキーバインド"
- "\n 特殊キー操作"
- "  c-f，control キーを押しながら     'f'"
- "  M-f，esc キーを押した後一度離して'f'"
- "    操作の中断 c-g，操作の取り消し（Undo） c-x u"
:license:
- "     cc by Shigeto R. Nishitani, 2016"
:cursor:
  :opts:
    :short: "-c"
    :long: "--cursor"
    :desc: Cursor 移動
  :cont:
  - c-f，move Forwrard,    前 or 右へ
  - c-b，move Backwrard,   後 or 左へ
  - c-a，go Ahead of line，行頭へ
  - c-e，go End of line,   行末へ
  - c-n，move Next line,   次行へ
  - c-p，move Previous line，前行へ
:page:
  :opts:
    :short: "-p"
    :long: "--ページ"
    :desc: Page 移動
  :cont:
```

```
- c-v, move Vertical,           次のページへ
- M-v, move reversive Vertical, 前のページへ
- c-l, centerise Line,          現在行を中心に
- M-<, move Top of file,        ファイルの先頭へ
- M->, move Bottom of file,     ファイルの最後尾へ

（以下省略）
```

1 行目の**:head:**の後ろの**:**を消してみる．**:**を消しただけで，メモを更新しようとすると

```
/Users/saki/.my_help% emacs_help --edit
/System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0/psyc
from /System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0
from /System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0
from /System/Library/Frameworks/Ruby.framework/Versions/2.0/usr/lib/ruby/2.0.0
from /Library/Ruby/Gems/2.0.0/gems/my_help-0.4.3/lib/specific_help.rb:17:in `i
from /Library/Ruby/Gems/2.0.0/gems/my_help-0.4.3/lib/specific_help.rb:12:in `n
from /Library/Ruby/Gems/2.0.0/gems/my_help-0.4.3/lib/specific_help.rb:12:in `n
from /Library/Ruby/Gems/2.0.0/gems/my_help-0.4.3/exe/emacs_help:4:in `<top (re
from /usr/bin/emacs_help:23:in `load'
from /usr/bin/emacs_help:23:in `<main>'
```

このようになり，emacs_help が動かなくなってしまう．使い慣れれば気づくことができるが，初心者ではエラーが見つけにくいと思われる．

#### 4.0.5 my_help から latex への変換

上図からも分かるように，本研究の my_help2hiki でも，my_help から latex へ変換するには一度 hiki に変換しなければ latex へ変換することができない．

#### 4.0.6 コマンド

my_help から hiki への変換，hiki から latex への変換の操作を行うコマンドの二つを行わなければ全てのファイルを同じ内容にすることはできない．

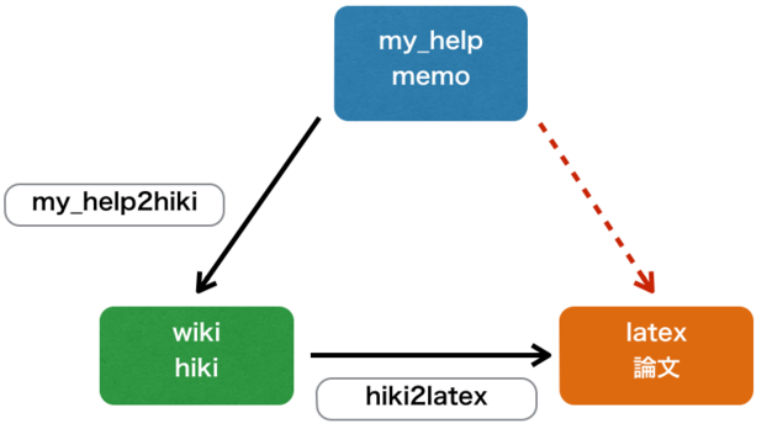


図 7　my_help2hiki による memo，hiki，latex の関係

#### 4.0.7　西谷研究室の内部サイトへの表示

西谷研究室には内部サイトがあり，研究室内で使うシステムのマニュアルなどが公開されている．hiki 形式への変換ができれば wiki で表示することはできるようになるが，my_help2hiki のコマンドで wiki のページを作成しても，現段階では研究室内全員が見ることはできない．

先ほど記述した欠点をふまえて，my_help2hiki に既存システムである hiki2latex を組み込み **my_help2hiki による memo，hiki，latex の関係**の図の破線部分にあたるシステムを作ることで，my_help から latex へとすぐに変換することのできるような，my_help，hiki，latex をさらに強く結びつけるシステムができるのではないかと考えている．また，西谷研究室には内部サイトがあり，研究室内で使うシステムのマニュアルなどが公開されている．hiki 形式への変換ができれば wiki で表示することはできるようになるが，my_help を hiki や latex に変換するのと同じように，その内部サイトに各研究室生が my_help によって作成した memo を表示できるコマンドを追加すれば，さらに便利なシステムになると考えられる．