卒業論文

Hikiのプラグインによる
レポート作成・投稿システムの開発

関西学院大学理工学部

情報科学科 西谷研究室

2511 澄田　紳弥

2016年3月

概 要

論文の執筆ツールとして LaTeX が広く利用されている．しかし, LaTeX 初心者にとっては扱いづらいものである．そこで, 体裁の整ったレポートを容易に作成するシステムの開発を目的に研究を進めた.

本研究では, 組版処理システムである LaTeX と, Wiki のクローンのひとつである Hiki を使用する．Hiki のコードはプログラミング言語 Ruby で記述されている．Ruby の特徴として, ソースコードをコンパイルすることなく実行できる点, ライブラリの導入が簡単である点が挙げられる．これにより, 効率良く Hiki のプラグイン開発ができる.

開発方法としては, Hiki のプラグインを開発し機能拡張することで, レポート作成ページ, 編集ページを作成した．また, hiki2latex.rb という Ruby のライブラリを用いて, Hiki 記法を LaTeX コマンドへ変換することで, レポート作成ページで執筆した文章を tex 形式のファイルとしてダウンロードできる機能を実装した.

結果的に, 執筆者はテキストフォームに文章を入力するだけで, 自分のレポートの tex 形式のファイルが得られるため, レポートの中身に専念することができる．また, 作成したレポートを Hiki 上で共有, 及び提出ができるようにレポート投稿システムを作成した．Hiki 上の操作だけでレポート作成から投稿までの, 一連の流れを実現することができた.

2 度のシステムリリースを通して, ユーザーから不具合の報告を受け, システムの不具合改善に加え, システムの問題点を発見することができた．これにより, システムが発展する可能性も十分見いだせた.

目 次

第1章　序論

1.1　研究目的

本研究の目的は, 体裁の整ったレポートを容易に作成するシステムの開発である. そのため, 本研究では, 組版処理システムである LaTeX と Wiki のクローンのひとつである Hiki を使用する. 入力フォーマットを Hiki のプラグインとして作成し, LaTeX と Hiki を連携することで, 文章の執筆に集中できるようなシステムの開発に取り組む. 入力フォーマットを用意することで, ユーザーは文章の執筆のみに集中でき, LaTeX コマンドへ変換することで, LaTeX で編集する手間を省くことができる. 文章の作成だけでなく, 文章の編集機能を実装することで, レポートを作成しやすい環境を整える.

1.2　研究背景

ワープロソフトとしては, Word が一般的であり, WYSIWYG（What You See Is What You Get）と表される通り, ディスプレイの表示と出力結果が同じであるため, 作業を行いやすい. しかし, 文章のレイアウトや, 数式や図・表の作成に時間が割かれるために, 文章の執筆に集中できない. また, 文章の量が増えるにつれ, 編集箇所が分からなくなったり, 編集する内容を思い返しながら作業しなければならないので, 編集作業の面でも文章執筆に集中することは難しい. 反対に, LaTeX はページ全体のレイアウトを整え, 複雑な表や数式を記述できる. しかし, LaTeX 初心者にとって, 参考書などで調べながらコマンドを扱うことは難しく, 肝心の文章に集中できないという問題点があった.

第2章　先行研究

2.1　既存システムの評価

2.1.1　Hiki

サイト構築や更新を手軽にできるようにHikiを利用する．HikiとはWikiのクローンの一つで, Wikiとの違いはコード がRubyで記述されていることである．また, CMSであるHikiは, Webブラウザから編集を簡単に行えるため, 多人数で作業を行うシステムの構築に適している．CMS(Contents Management System)とは, HTMLなどを習得することなく, Webサイトを構築・編集できるソフトウェアである．また, Hikiの特徴には,

- オリジナルWikiに似たシンプルな書式
- プラグインによる機能拡張
- 出力するHTMLを柔軟に変更可能
- アクセス制限が可能
- システムのプロトタイプを容易に作成可能
- ページを容易に追加, 編集可能

が挙げられる [1].
Hikiのページ編集はHiki記法による記述で行われる．HikiDocというライブラリが用いられているので, Hiki記法で書かれたテキストを HTML に変換できる．Hiki記法はHTMLよりシンプルなので, 入力フォーマットの作成には適している．例えば, 見出しは「!」, 箇条書きは「*」などで記述できるので, 直感的な理解が容易である．これらの特徴を利用し, 本研究では, Hikiをプラグインによって機能拡張し, レポート作成・投稿システムを作成する．

2.1.2 LaTeX

LaTeX とは, 美しい文書を簡単に作成することができる優れた組版ソフトである. 数式の処理に優れ, レポートや論文の作成に有効なフリーソフトウェアとして知られている. 特に, 理系の論文や本の製作に広く使われている [2]. LaTeX には,

- レイアウトを整形
- 改行や空白を制御
- ページや章構成を管理
- 表や数式を記述
- 目次や参考文献の一覧などを作成

などの特徴が挙げられる [3].
ワープロソフトで有名な Microsoft Word のような, ディスプレイに表示されるものと処理結果が一致するようなソフトは, ユーザーにとって作業しやすいというメリットがある. これは, 初心者にとっては利用しやすいものだが, 製版するための細かい作業をするには不向きである. 全体のレイアウトを構成しなければならないので, 膨大な文章を編集する場合に, 執筆に集中することができなくなる. LaTeX は原稿をプレーンテキストで作成するので, 編集しているファイルを出力結果に違いがあるので, 初心者には扱いづらい. しかし, 上記の特徴から, LaTeX では論理構造が整った論文が作成できる. よって, LaTeX は習得するまで時間はかかるが, 文章の編集には適している.

2.2 TeX 用編集エディタの選定

編集用エディタとして TeXShop を用いる. TeXShop は OS X で動作する TeX 編集用エディタであり, LaTeX コマンドを編集, 実行できる. LaTeX 文書編集ツールとして既に存在する LyX, Cloud LaTeX と比較することで, TeXShop の優位性を検証する. LyX とは, Matthias Ettrichniyotte によって開発された文書プロセッサである. 現在は, The LyX Team によって開発が進められている. TeX をバックグラウンドで使用するので, 数学的文書の作成や, 学術的文書や論文, 書籍などの構造化された文書作成において使用される

[7]. Cloud LaTeX とは, 株式会社アカリクの Cloud LaTeX development Team によって開発された, ブラウザ上で LaTeX 文書を作成・編集し, 自動コンパイルができるサービスである. Cloud LaTeX をマルチバイト言語へ対応することで, 日本・世界における研究者・学生のためのサービスとして注力されている [8]. これら 2 つとの比較結果を表 2.1 にまとめた. 以下で, それぞれの性能を評価する.

表 2.1: TeXShop と既存 TeX サービスの比較表.

	TeXShop	LyX	Cloud LaTeX
タグ移動	○	○	×
日本語マニュアル	×	○	×
検索・置換	○	○	×
動作速度	○	×	○
分割表示	○	○	×
変更追跡	×	○	×

- Cloud LaTeX はローカルに LaTeX 環境を準備しなくても, Web 上で LaTeX を書くことができる. また, コマンド編集をすると, ほぼ同時にプレビューに反映されるのでユーザーにとっては作業がしやすい環境である. しかし, 不安要素として, 操作メニューが乏しいことや, テンプレートはあるが変更に手間がかかることが挙げられる.

- LyX は文章構造の設定や, 図表や数式の挿入は簡単にできるが, 操作メニューの数が多いので, 膨大なマニュアルが必要になっている. また, 文章量が多くなると, 動作速度がかなり遅くなるので, 使い勝手が悪い.

- TeXShop はコマンド入力をしなければならないが, マクロといったコマンド入力のためのサポート機能は備わっており, 動作速度も安定している. また, タグ機能があり, ページをスクロールすることなく編集したい項目に移動できる.

以上の結果より, 柔軟性と動作速度が安定している TeXShop を選定する.

第3章　開発手法

従来の論文執筆方法は, 編集エディタに LaTeX コマンドを打ち込み, 実行することで pdf を出力する方法である. この場合, LaTeX コマンドを調べながら執筆するので, 文章自体に集中できないという問題点があった. そこで, 本研究で開発したシステムは以下の機能が要求される.

- Hiki のページ上に入力フォーマットを作成
- Hiki 記法から LaTeX コマンドへの変換

従来の執筆方法と今回提案するシステムを利用した執筆方法を図 3.1 で示した. 入力フォーマットを作成することで, ユーザーの論文のレイアウトを構成する手間を省くことができる. また, Hiki 上の入力フォーマットで書かれた文章が LaTeX コマンドへ自動変換されるので, LaTeX 初心者でもコマンドを調べることなく文章の執筆に集中できる.

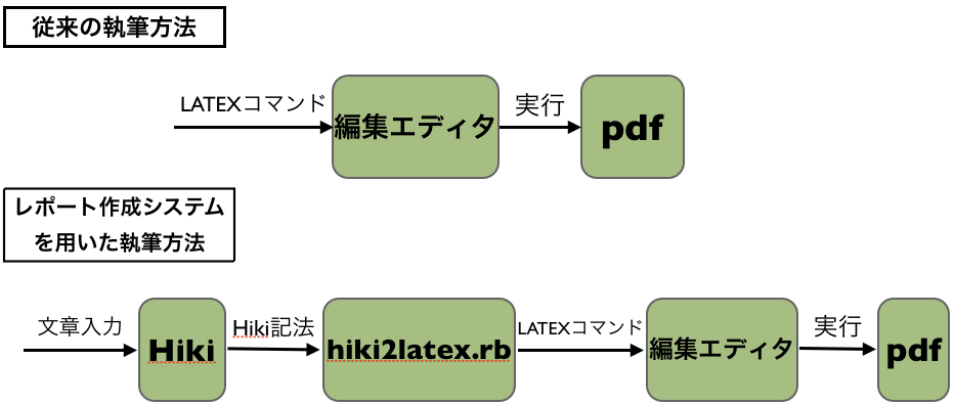

図 3.1: 従来の執筆方法とレポート作成システムを用いた執筆方法の比較.

3.1 LaTeX コマンドへの変換方法

hiki2latex.rb という Ruby のライブラリを用いることで, Hiki 記法を LaTeX コマンド変換する. TeX へ自動で変換するツールは多く開発されており, 今回は表 3.1 で一部の既存 TeX 変換ツールと比較する [4, 5, 6].

表 3.1: hiki2latex と既存 TeX 変換ツールの比較表.

	hiki2latex	Word2TeX	Excel2LaTeX	HTML2TeX
テキスト	○	○	×	○
図	○	○	×	×
表	○	○	○	×
数式	○	○	×	△
参考文献	×	○	×	△

以下で, それぞれの性能を評価する.

- HTML2TeX は簡易な数式しか処理できず, 図や表の変換は不可である.
- Excel2TeX は表の変換には特化しているが, 他の項目に関しては, 変換は不可である.
- hiki2latex と Word2TeX については, 両方ともに変換としての機能は十分である.

上記の比較結果より, Word2TeX と hiki2latex が十分な変換機能を備えている. しかし, Word はワープロソフトなので, Web アプリケーションの開発に用いるのは適していない. よって, 今回は Web アプリケーションの開発を目的としているので, Hiki を利用した hiki2latex をレポート作成・投稿システムに組み込む.

また, 上記の表 3.1 で比較に挙げた項目を対象に, Hiki 記法と LaTeX コマンドの対応を表 3.2 に示す. 表 3.2 でわかるように, 執筆量の多い LaTeX コマンドと比べ, Hiki 記法は簡単な記号で書かれるため, ユーザーへの負担は少ない. よって, レポート作成システムに要求される機能の一つを満たしている.

表 3.2: Hiki 記法と LaTeX 命令の対応表.

	Hiki 記法	LATEX コマンド
目次	{{toc}}	\tableofcontents
著者	!author:著者	\author{著者}
表題	!title:表題	\title{表題}
見出し	!見出し	\section{見出し}
箇条書き	*箇条書き	\begin{itemize} \item 箇条書き \end{itemize}
テーブル	||300||200||	\begin{table}[htbp]\begin{center} \caption{} \begin{tabular}{lll} \hline 300 &200 & \\ \hline \hline \end{tabular} \label{default} \end{center}\end{table}
画像添付	{{attach_view(hoge.png)}}	\begin{figure}[htbp]\begin{center} \includegraphics[width=6cm]{./hoge.png} \caption{} \label{default}\end{center}\end{figure}
数式	{{dmath 'f_x'}}	\begin{equation} f_x \end{equation}

第4章　開発したシステム

4.1　レポート作成システム

4.1.1　仕様

レポートを作成するためのシステムである. レポート作成システムに要求される機能は以下の通りである.

1. 入力フォーマットの表示
2. レポートの作成
3. txt 形式ファイル及び tex 形式ファイルの出力
4. レポートの編集
5. 3-4 を繰り返し実行

入力フォーマットを表示し, ユーザーは文章を執筆する. レポート作成後には,tex 形式ファイルと txt 形式のファイルのダウンロードリンクが設置されているため, ダウンロード可能である. また, 作成した文章をデフォルトとしたテキストエリアが設置されているので, そこでユーザーは文章の編集が可能である. 何度でも編集が可能であり, 編集される度に最新のファイルが出力される.

4.1.2　動作

図 4.1 はレポート作成システムの一連の流れを， 振る舞いにおいてオブジェクト群がどのようにコラボレーションを行うか表すシーケンス図を用いて示している [9]. ユーザーは, レポート入力ページにレポートを入力し, 投稿ボタンを押下する. 入力データは, レ

ポート編集ページに渡される．レポート編集ページで, ダウンロードリンクを押下することで, txt 形式ファイルと tex ファイル形式がダウンロードできる．また, レポートを編集すると, 編集後のレポート編集ページが作成され, 編集後のファイルをダウンロード可能である．よって, 編集は何度でも実行でき, その度に, 最新のファイルをダウンロードできる．

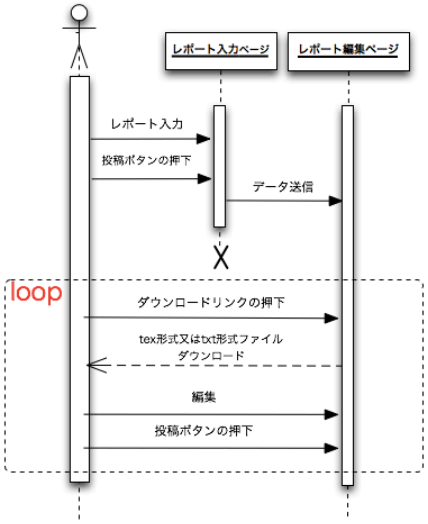

図 4.1: レポート作成システムのシーケンス図.

4.1.3 操作手順

レポート作成システムの操作手順を示す．図 4.2 は, 入力ページのフォーマットを示している．入力する項目をページ名, 表題, 著者, 目的, 先行研究, 資料と方法, 結果と分析, 考察, 結論として作成した．ページ名, 表題, 著者については必須入力項目とし, それら以外の項目は任意入力項目とする．任意入力項目に対しては, それぞれに小節と小々節を作成した．また, テンプレートの任意入力項目名は変更可能にしている．任意の項目のチェックボックスにチェックを付け, 変更があれば項目名を変更する．そして, テキストエリア内で文章を執筆し, 投稿ボタンを押下する．

図 4.2: レポート作成ページ.

レポート作成後のページを図 4.3 で示す. レポート作成後のページでは, 文章の編集, txt 形式ファイルと tex 形式ファイルのダウンロードが可能である. 文章の編集をする場合は, 執筆した文章がデフォルトで設定されているテキストエリアで行う.

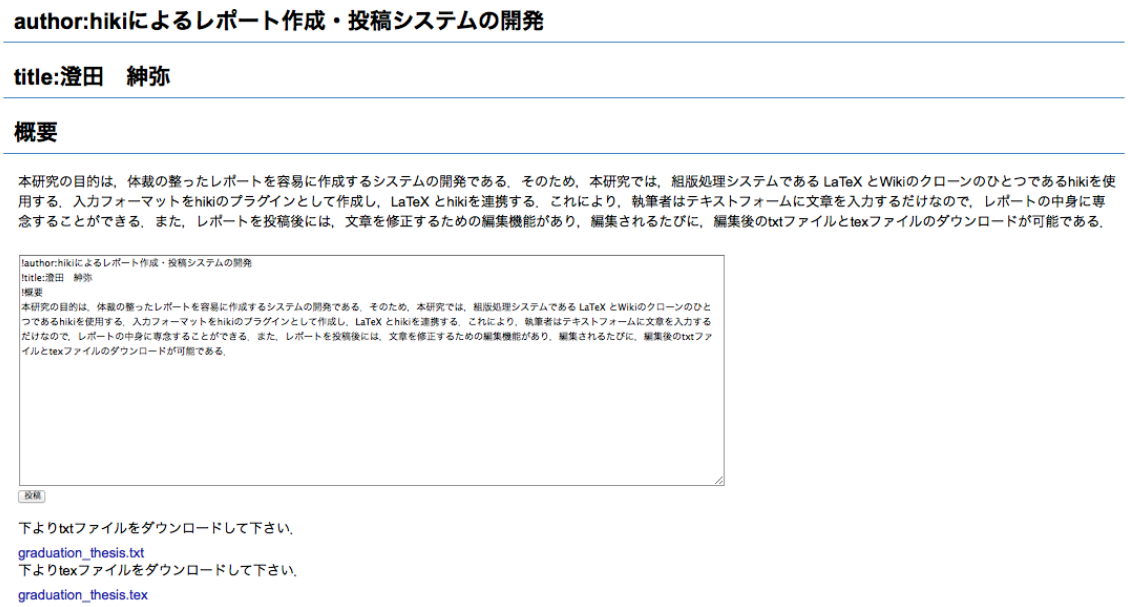

図 4.3: レポート作成後ページ（編集前）.

レポート編集後のページを図 4.4 で示す．編集後のページでは, 編集前と編集後の差分が出力される．編集するたびに, 最新のファイルがダウンロード可能である.

図 4.4: レポート作成後ページ（編集後）.

4.2　レポート投稿システム

4.2.1　仕様

ユーザーがレポートを投稿するためのシステムである. レポート投稿システムは以下の機能が要求される.

1. 投稿者及び添付されたファイルの表示
2. 投稿者及びファイルの選択
3. レポートを投稿

投稿者とファイルの表示及び選択機能を実装している. ページにファイルが添付されると, ファイル選択欄に, 添付したファイルが表示される. ユーザーは, 投稿者とファイルを選択し, 投稿ボタンを押下することで, 投稿が完了になる.

4.2.2　動作

図 4.5 はレポート投稿システムのシーケンス図である. ユーザーは, 投稿ページで, 投稿者及びファイルを選択する. その後, 投稿ボタンを押下することで, 投稿先ページにファイルが送られる.

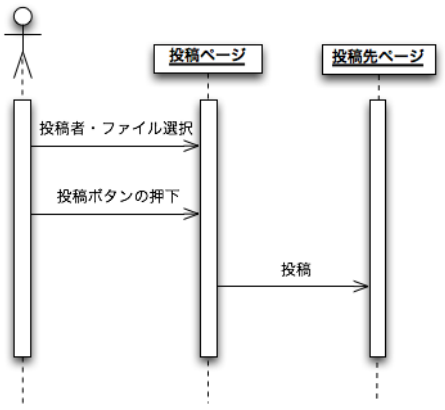

図 4.5: レポート投稿システムのシーケンス図.

4.2.3 操作手順

レポート投稿システムの操作手順を示す. 図 4.6 は, ファイル添付前の画面である. 投稿ページにアタッチフォームを用意しているので, ユーザーはファイルを添付する.

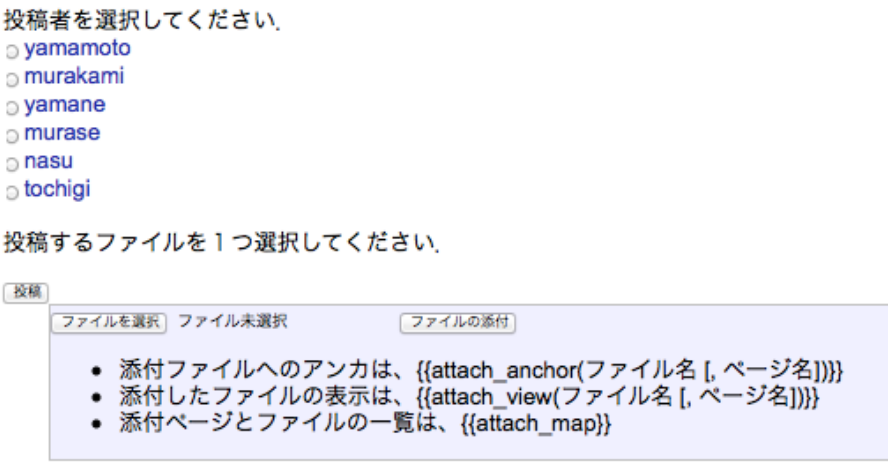

図 4.6: 投稿ファイル添付前.

図 4.7 は, ファイル添付後の画面である. ファイルを添付後, 投稿者とファイルのラジオボタンをそれぞれ選択し, 投稿ボタンを押下することで, 投稿が完了する.

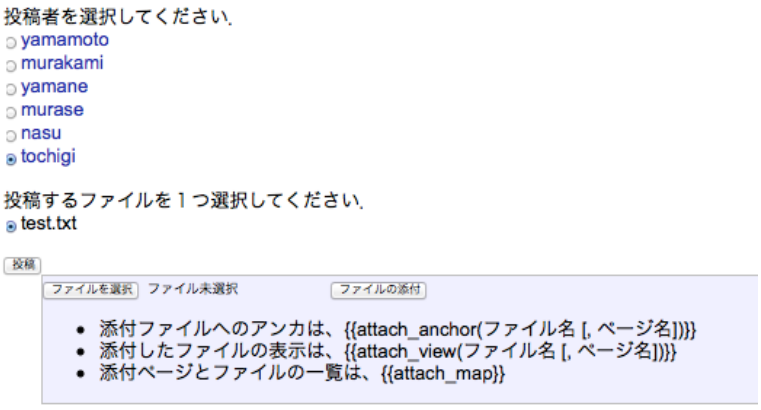

図 4.7: 投稿ファイル添付後.

第5章　実装

5.1　使用プログラミング言語Ruby

プラグインの作成言語はRubyである．Rubyは，まつもとゆきひろが開発したオブジェクト指向スクリプト言語である． クラス，継承，メソッドなどのオブジェクト指向言語の機能を持ち， コンパイルすることなくプログラムを生成し， 実行可能である． また， 仕様の１つに動的型付けがあるため，あらゆるデータの型を変数に格納できる．また，宣言無しに変数を使用できる．ストレスなくプログラムをすることを前提に作られているので，複雑な記述方法などが軽減されており， プログラムの内容に集中できる．

5.2　レポート作成システムのプラグイン

レポート作成システムのために，report_generation.rb と edit.rb を開発した．これらのプラグインの挙動を図 5.1 に示した．

report_generation.rb は comment.rb を参考に開発したプラグインである．comment.rb はページ中の任意の場所へ配置し，1 行コメントを入力するためのフォームを生成するプラグインである [10]．report_generation.rb に以下の機能を実装した．

- 入力フォーマットの表示
- レポートの作成
- tex 形式ファイルと txt ファイルの出力

関数 report_generation で入力フォーマットは表示されるので，ユーザーはそれぞれの任意入力項目に対して用意されたテキストエリアに文章を入力する．文章を入力し，投稿ボタンの押下を実行すると，関数 report_generation_post が実行され，入力された文章をそれぞれのファイル形式で出力し，レポート作成後のページができる．

edit.rb は作成した文章を何度でも編集可能にするプラグインである．以下の機能を実装した．

- レポートの編集
- 編集後の tex 形式ファイルと txt ファイルの出力

関数 edit で編集するためのテキストエリアが表示される．そこには編集前の文章がデフォルトで入力されている．また, 編集するたびに, 最新の txt 形式ファイルと tex 形式ファイルをダウンロードすることができる．

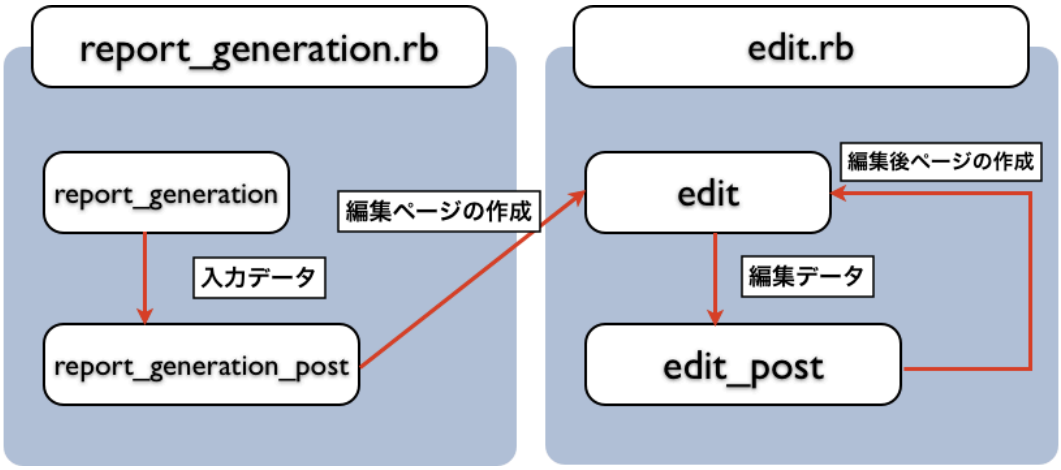

図 5.1: レポート作成システムに用いたプラグインの挙動.

5.3 レポート投稿システムのプラグイン

レポート投稿システムのために, submitform.rb を開発した．このプラグインの挙動を図 5.2 に示した．

submitform.rb は選択された投稿者及びファイルを別ページへ移動するためのプラグインである．以下の機能を実装した．

- 投稿者とファイルの表示
- レポートの投稿

関数 submitform では, 予め登録されている投稿者と添付されたファイルを表示する. 投稿ボタンが押下されると, 関数 submitform_post が実行され, 投稿が完了する.

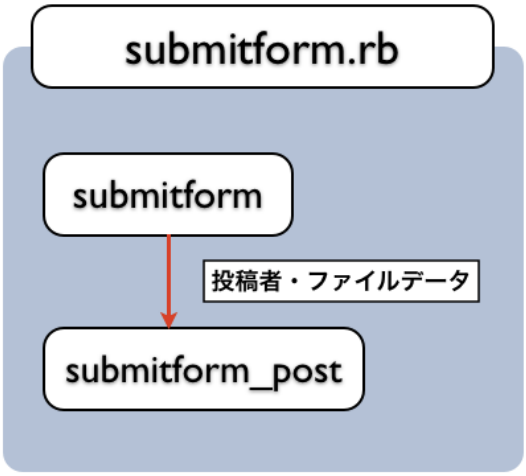

図 5.2: レポート投稿システムに用いたプラグインの挙動.

第6章　ユーザーからのフィードバック

6.1　システムリリース

6.1.1　１回目のリリース結果及び考察

1回目のリリース時には, 以下の不具合が発生した.

- 文章が長過ぎるため, 作成できない.

この原因は, HTML で form 要素の method 属性をデフォルトにしており, GET メソッドを使ってしまっていたため, URL が長過ぎて, 投稿できないというものであった. よって, POST メソッドを指定して, 送信することで改善できた. これは, 十分なテストを行わずリリースした結果, 生じた不具合であった.

6.1.2　２回目のリリース結果及び考察

２回目のリリース時には, レポートの作成及び投稿ができた. しかし, 以下の不具合が発生した.

- 一部のレポート項目の内容が反映されない.

この原因は, 単にプログラム内での変数間違いであった. 単純な間違いであったが, 入力ページのテキストフォーム, テキストエリアが多いため, その分プログラムが煩雑になっていたのが原因であった. この問題から, クラス化もしくはハッシュを用いることができれば, 作業効率は向上し, 事前にミスも防げる.

また, ２回のシステムリリースを通して, ユーザーからシステムの操作手順がわからないというフィードバックがあった. 解決策としては, マニュアルの作成をした. しかし, これではユーザーへの負担が増えるため, 根本的な解決にはならない. CSS(Cascading Style

Sheets) を編集してスタイルをデザインし, Javascript を用いて動的な Web ページを作成することで, さらに直感的な操作を可能にすることができる.

第7章　総括

7.1　本研究による成果

本研究で得られた成果をまとめると以下の通りである.

- 入力フォーマットの作成. フォーマットに沿って文章を執筆することで, 文章レイアウトの構成に手間がかからなくなった. レポートを課す人によって, レポートの体裁は異なる可能性がある. しかし, Web ページ上にフォーマットが用意されているので, 全てのレポートの体裁を統一できる.
- LaTeX コマンドへの自動変換. 入力した文章が tex 形式のファイルとして出力されるので, LaTeX コマンドを調べながら執筆する必要がなくなった.
- 別ページへのレポート投稿機能を実装したことで, ファイルの共有が可能になった.

入力フォーマットを Hiki のプラグインで作成し, 入力した文章を LaTeX コマンドへ変換することで, 体裁の整ったレポートを容易に作成することができ, ユーザーへの負担は軽減され, 文章のみに集中できる環境ができた.

7.2　残された問題点

現段階で以下の問題点が残る.

- 図や表, 数式を記述したい場合には, 直接 Hiki 記法で書かなければならない.
- 作成したプラグインは関数のみを使っていて, 煩雑な記述になっている.

直接 Hiki 記法で書かなければならないので, ユーザーへの負担が大きい. Hiki 記法で入力するためにサポート機能を実装しなければならない. 例えば, 数式というタブを用意し, タ

ブを押下することで, 数式の Hiki 記法のテンプレートが表示されるような機能が必要である．また, プログラムが関数だけで開発しており煩雑になっているため, 編集する箇所を探すのに時間がかかり, 変更箇所が多かったため, 作業効率が悪かった．今後の発展を見据えて, クラス化やハッシュを使用することで編集しやすいプログラムにする必要がある.

謝辞

本研究を進めるにあたり, 様々なご指導を頂きました西谷滋人教授先生に深く感謝いたします. 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] 「Hiki」, たけうちひとし, http://hikiwiki.org/ja/about.html, 2016/1/24 アクセス.

[2] 「LaTeX2 ε 美文書作成入門」, 奥村晴彦, 黒木裕介, (技術評論社 2013).

[3] 「CNS GUIDE 2002」, CNS ガイド編集委員会, http://cns-guide.sfc.keio.ac.jp/2002/13/1/1.html, 2016/1/24 アクセス.

[4] 「Chikrii Softlab: Word to LaTeX, LaTeX to Word Converters」, Chikrii Softlab, http://www.chikrii.com/products/word2tex/features/, 2016/2/4 アクセス.

[5] 「Excel2LaTeX マニュアル」, 中尾誠, http://sci-tech.ksc.kwansei.ac.jp/~inagai/dictionary/TeX/Manual.html, 2016/2/4 アクセス.

[6] 「HTML to LaTeX」, Kenji Arisawa, http://ar.aichi-u.ac.jp/blog/html2tex/html2tex.html, 2016/2/4 アクセス.

[7] 「LyX」, The LyX Team, https://www.lyx.org/WebJa.Home, 2016/2/2 アクセス.

[8] 「Cloud LaTeX produced by Acaric」, 株式会社アカリク, https://acaric.co.jp/acaric-cloudlatex/, 2016/2/4 アクセス.

[9] 「UML モデリングのエッセンス 第 3 版」, Martin Fowler, 羽生田栄一, (翔泳社 2005).

[10] 「Hiki」, たけうちひとし, http://hikiwiki.org/ja/comment.rb.html, 2016/2/7 アクセス.

付 録A　コード解説

A.1　レポート作成システムのプラグイン

レポート作成システムに用いた, report_generation.rb と edit.rb についてコード解説をする.

A.1.1　report_generation.rb

入力ページは関数 report_generation で表示され, 入力フォーマットは, HTML で記述する. 下記は HTML で記述した入力フォーマットを一部抜粋したものである.

```
<head>
<script>
function chkdisp( obj,id ) {
if( obj.checked ){
   document.getElementById(id).style.display = "block";
}
else {
   document.getElementById(id).style.display = "none";
}
}
</script>
</head>

<form action="#{@conf.cgi_name}" name="myform" method="post">

<br><input type="checkbox" id="chk1" onclick="chkdisp(this,'ans1')">目的
<p id="ans1" style="display:none;">
<input type="text" name="purpose" size="10">
<br><textarea name="purpose_textarea"
id="purpose_textarea" cols="60" rows="10"><textarea>
<br>      <input type="checkbox" id="chk7" onclick="chkdisp(this,'ans7')">小
節</p>
```

また, チェックボックスによるテキストエリアの表示, 非表示は javascript で記述する. フォーム内でチェックボックスがチェックされると, onclick で関数 chkdisp が作動する. 関数 chkdisp では, getElementById() を利用して, ID 属性値から要素を取得し, display で要素の表示形式を指定する. もしチェックボックスにチェックがされれば,display の値は block となり表示され, チェックされなければ, display の値は none となり非表示になる. 投稿ボタンを押下後に関数 report_generation のデータを関数 report_generation_post で受け取る. 関数 report_generation_post では, 受け取った入力データを基に, txt 形式ファイルと tex 形式ファイルを作成し, ダウンロードリンクを設置する.

```
  <input type="submit" name="comment" value="#{comment_post_label}"></br>
  <input type="hidden" name="c" value="plugin">
  <input type="hidden" name="p" value="#{h(@page)}">
  <input type="hidden" name="plugin" value="report_generation_post">
```

上記で, 関数 report_generation_past へデータが渡される.

```
if params['purpose_textarea'].empty? == false
   if params['purpose'].empty? ==false
      content << "!#{purpose}\n"
      content << "#{purpose_textarea}\n"
   end
   if params['purpose'].empty? == true
      content << "!目的\n"
      content << "#{purpose_textarea}\n"
   end
end
```

まず, 関数 report_generation のフォームで入力した値を params[' パラメータ名'] で取得する. そして, content に入力したデータを代入する. content は String 型変数で, 後の記述で使用する.

```
FileUtils.mkdir_p("#{home}/Sites/hiki-1.0/data/cache/attach/#{page}")
unless FileTest.exist?("#{home}/Sites/hiki-1.0/data/cache/attach/#{page}")
test=File.open("#{home}/Sites/hiki-1.0/data/cache/attach/#{page}/#{page}.txt"
,"w")
test.puts content
test.close
```

~cache/attach/#{page}へ txt 形式ファイルを作成する. #{page}はフォームのページ名の項目で入力された文字列であり, レポート作成後のページで txt 形式ファイルのダウンロードが可能になる. #{page}.txt に content を代入されていた値を書き込む. tex ファイルについては, 上記で作成された, txt ファイルを参照し, hiki2latex.rb を実行する.

```
tmp=open("#{home}/Sites/hiki-1.0/data/cache/attach/#{page}/#{page}.txt","r")
result = hiki2latex(tmp)
tmp1=open("#{home}/Sites/hiki-1.0/data/cache/attach/#{page}/#{page}.tex","w")
```

hiki2latex.rb は, Hiki 記法で書かれた文章を, tex コマンドへ変換するプラグインである. レポート作成後のページで tex ファイルが作成され, ダウンロード可能になる.

```
content1 << "\n 下より txt ファイルをダウンロードして下さい. \n"
content1 << "\n{{attach_anchor(#{page}.txt)}}\n\n"
content1 << "\n 下より tex ファイルをダウンロードして下さい. \n"
content1 << "\n{{attach_anchor(#{page}.tex)}}\n\n"
test=File.open("#{home}/Sites/hiki-1.0/data/text/#{page}", "w")
test.puts content1
test.close
```

~data/text/#{page}に content1 に代入されていた値を書き込む. {{attach_anchor(ファイル)}}は, hiki 記法の一種で, ~cache/attach 内のファイルのダウンロードリンクを設置する. これらの記述で, レポート作成, レポート作成後ページでのファイルのダウンロードが可能になる.

A.1.2 edit.rb

投稿した文章を何度でも修正可能にするプラグインである. 編集するためのテキストエリアには, 編集前の文章がデフォルトで入力されている. また, 編集するたびに, 最新の txt 形式ファイルと tex 形式ファイルをダウンロードすることができる. diff_lcs.rb により, 編集前と編集後の差分も出力される. 関数 edit(page_name) では, 作成したレポートの表示と編集するためのテキストエリアの設置である. page_name はレポート作成後のページ名を引数としている.

```
content << "<form action=\"#{@conf.cgi_name}\" method=\"post\">"
test = File.open("#{home}/Sites/hiki-1.0/data/cache/attach/#{page_name}
/#{page_name}.txt","r")
test.each do |tmp|
      content1 << "#{tmp}"
end
test.close
content << "<br><textarea name=\"edit\" id=\"edit\" cols=\"50\"
rows=\"20\">#{content1}</textarea>"
```

まず, HTML で記述することで, 作成したレポートと編集するためのテキストエリアを表示する. tmp に#{page_name}.txt の中身を書き込み, content1 に代入することでレポートを表示する. また. デフォルト値を#{content1}にすることで, テキストエリア内に文章を表示する. 以下の記述で, 投稿ボタンを押下後に, 関数 edit_post が実行される.

```
<input type="submit" name="submit" value="#{comment_post_label}">
<input type="hidden" name="c" value="plugin">
<input type="hidden" name="p" value="#{h(@page)}">
<input type="hidden" name="plugin" value="edit_post">
<input type="hidden" name="session_id" value="#{@session_id}">
```

関数 edit_post では, 連続編集するための毎回保存と編集前と編集後の差分を出力する. 毎回編集するための記述を, 以下に示す.

```
content << "{{edit(#{page_name})}}\n"
save( @page, content1, md5hex )
```

content に代入されている値が, Hiki ページに記述される. これにより, 何度も編集後の画面に遷移される. また, 編集前と編集後の差分を出力するために, diff_lcs.rb を使う. これは, 2 つのファイルの差分を出力するプラグインである. まず関数 edit(page_name) で差分出力用のファイルを作成する.

```
from =  "#{home}/Sites/hiki-1.0/data/cache/attach/#{page_name}
/#{page_name}.txt"
to = "#{home}/Sites/hiki-1.0/misc/plugin/predit.txt"
FileUtils.cp(from, to)
```

FileUtils はファイル操作をするためのライブラリである, 今回は, FileUtils.cp(from,to) というモジュールを使うことで, from のファイルを to にコピーする. 関数 edit_post でも同様に作成する.

```
from =  "#{home}/Sites/hiki-1.0/data/cache/attach/#{page_name}
/#{page_name}.txt"
to = "#{home}/Sites/hiki-1.0/misc/plugin/edited.txt"
FileUtils.cp(from, to)
```

差分の出力方法は以下に示す.

```
predit=‘cat #{home}/Sites/hiki-1.0/misc/plugin/predit.txt‘
edited=‘cat #{home}/Sites/hiki-1.0/misc/plugin/edited.txt‘
diff_lcs(predit,edited)
```

predit,edited にファイル内の値を代入する. シングルコーテーションで囲むことで, 外部コマンドを実行する. また cat はファイル内を表示するコマンドである. そして, diff_lcs(predit,edited) で 2 つのファイルの差分を出力する.

A.2 レポート投稿システムのプラグイン開発

レポート投稿システムに用いた, submitform.rb についてコード解説をする.

A.2.1 submitform.rb

関数 submitform では投稿者とファイル名の表示が実行される. 投稿者の選択について, 以下に示す.

```
user = File.open("#{home}/Sites/hiki-1.0/misc/plugin/member.txt", "r")
user.each do |tmp|
      content << "<input type=\"radio\" name=\"member[]\" value=\"#{tmp}\">
<a href=\"./?#{tmp}_submit\">#{tmp}</a><br>"
end
test.close
```

予め, member.txt 内にユーザーの名前を書き込んでおく. ファイル内の書き込みを変更するだけなので, 容易にユーザー名を変更できる. `user.each do |tmp|` で member.txt 内の各行を読み込み, content に代入することで表示される. 次に, ファイルの選択について, 下に示す.

```
if File.exists?("#{home}/Sites/hiki-1.0/data/cache/attach
/#{page_name}") != false then
submit = Dir::entries("#{home}/Sites/hiki-1.0/data/cache/attach
/#{page_name}")
submit.each do |tmp1|
if tmp1 != "." && tmp1 != ".." then
      content << "<input type=\"radio\" name=\"file[]\" value=\"#{tmp1}\">
#{tmp1}<br>"
end
end
end
```

添付されたファイルを読み込み, HTML 表記で記述することで, 添付したファイルが表示される. そして, 以下の記述で, 投稿ボタンの押下後に, 関数 submitform_post が実行される.

```
<input type="submit" name="comment" value="#{comment_post_label}"></br>
<input type="hidden" name="c" value="plugin">
<input type="hidden" name="p" value="#{h(@page)}">
<input type="hidden" name="plugin" value="submitform_post">
<input type="hidden" name="session_id" value="#{@session_id}">
```

関数 submitform_post では, 投稿の処理が行われる.

```
member = params['member[]']
file = params['file[]']
```

上記で, 関数 submitform で選択された, データを取得する.

```
from = "#{home}/Sites/hiki-1.0/data/cache/attach/submitform/#{file}"
to = "#{home}/Sites/hiki-1.0/data/cache/attach/#{member}_submit/#{file}"
FileUtils.mv(from, to)
```

FileUtils.mv(from, to) というモジュールを使い, ファイルを from から to のパスへ移動する. これにより, 投稿前ページのファイルが投稿後のページへ移動する事ができる. これらにより, 投稿者とファイルの選択から投稿するまでの処理が実行可能になる.