1B4-2

オフラインWeb技術に基づくPDF文書添削システムの実現

A System for Annotating PDF Document Based on the Offline Web

合田拓史[*1] 片山真也[*2] 白松俊[*2] 大囿忠親[*2] 新谷虎松[*2]
Takushi Goda Shin-ya Katayama Shun Shiramatsu Tadachika Ozono Toramatsu Shintani

[*1]名古屋工業大学 工学部 情報工学科
Dept. of Computer Science, Nagoya Institute of Technology

[*2]名古屋工業大学大学院 工学研究科 情報工学専攻
Dept. of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology

We aim to develop a PDF annotation system for touching up papers. Although the existing system we developed had functions for reading documents, adding annotations to documents, and management documents, the system did not support to touch up papers. We postulate a model to correction of papers, state some problem on it. We propose a method for building a support system to touch up papers. The result shows how the system can be effectively used for touching papers.

1. はじめに

本研究では，主にPDFファイルを対象とした資料の管理・閲覧・情報の付加を行うシステムを開発している [合田 13]．現在，本システムを学生・教員間での論文添削を目的として利用している．本システムはWebアプリケーションとして実装しており，複数人での閲覧・添削状況の同期機能を持ち，被添削者がリアルタイムに添削状況を見ることができる．しかし添削内容に関しては具体的な支援を行っておらず，論文添削の場を提供するにとどまっている．本研究では，システムで論文の添削に関して支援を行うための手段について検討を行った．本稿では，論文添削サイクルの具体的なモデルを仮定し，そのモデルにおいて論文添削作業における具体的な支援方法を提案する．また，それらの実現における課題，課題を解決するためのアプローチについて述べる．

2. 論文添削における課題

2.1 論文添削のモデル

本研究における論文添削のモデルとして，図1のモデルを仮定する．まず，論文の添削作業は修正指示を行う添削者と，論文の修正を行う被添削者によって，次のようなサイクルで行われると仮定する．まず最初のプロセスとして，被添削者が添削者に対して論文の提出を行う．本研究で開発しているシステムにおいては論文PDFのアップロードがこれにあたる．2つ目は添削プロセスで，添削者が提出された論文に対して修正の指示を与えるプロセスである．実際に印刷した論文を用いての添削作業では，口頭での指示や論文に直接書き込む等の方法が用いられるが，本システムでは文書に対してアノテーションを付加することで行う．3つ目は修正プロセスで，与えられた指示に応じて被添削者が論文に修正を加えるプロセスである．本システムにおいては，論文のPDF自体に対して操作を行わないため，添削によりアノテーションが付加された資料を示すことでのみ支援を行い，実際の修正論文の作成は外部のシステムを利用する事となる．その後，必要であれば再度修正した論文の提出を行う．本研究では論文の添削作業をこのようなモデルと仮定する．以降で，このモデルにおいて本システムを用いて行われる論文添削の課題について述べる．

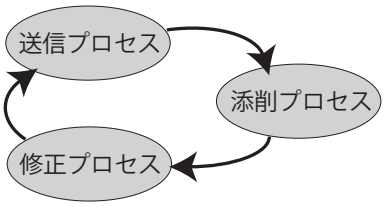

図 1: 論文添削のモデル

2.2 論文の修正情報の検出

論文の添削において，添削者が行った添削，すなわち修正指示に対して，被添削者がどのように修正を行ったかを知る必要がある．実際の添削では，修正前後の論文を比較し修正箇所を探すことで行われる．この作業は人手で行われるため，探す手間がかかることや確認漏れ等の問題が存在する．これをシステムで支援できれば添削者の負担を軽減でき，確認漏れを減らすことが可能であると考えられる．論文添削において，添削開始直後以外は全体を書き直すということは少ないと考えられる．多くの変更を行う場合でも重要な部分や大きな変更部分の指示を先に行い，被添削者によってそれらが修正された後で細かい部分の添削に移るというかたちで行われる場合が多い．また，編集による差分情報を元の資料に適用することができれば，Webアプリケーションとしての通信効率の改善や，システムとして保持するデータサイズの削減等のパフォーマンス向上も期待できる．

2.3 添削・修正状況の通知

添削においては，添削者と被添削者の間で論文の交換を行いながら行われる．そのため，論文の添削が終わればそれを被添削者に通知する必要があり，修正が終われば添削者に通知する必要がある．実際の添削では，メール等を用いて通知を行う場合が多いが，添削の度にこれを行うのは手間となる．本研究において開発しているシステムにおいても，添削・修正状況の通

連絡先: 合田拓史，名古屋工業大学 工学部 情報工学科，〒466-8555 愛知県名古屋市昭和区御器所町，052-733-6550，godata@toralab.org

知に関する支援は行っておらず，同様にメール等を用いた通知を添削者・被添削者が行う必要がある．これをシステムによって支援できれば，添削における負担を軽減することができると考えられる．

2.4 添削内容の競合

本研究における論文の添削では，添削者が複数存在する事を想定している．複数の添削者が一つの論文に対して添削を行うため，添削内容の競合が発生する事が十分考えられる．この場合の競合とは，同一の箇所に対して複数の添削者が添削を行う事である．この場合，競合として次の３つが考えられる．１つ目は，同一箇所に対して同様の添削が行われた場合，すなわち添削内容の重複である．２つ目は，同一箇所への添削が行われたが，それらが無関係な場合である．３つ目は，矛盾する添削を行った場合であり，複数の添削者が同一箇所に対して行った添削，すなわち修正指示の両立が不可能な場合である．これら競合した添削の解消を行うことができれば，被添削者の負担軽減や論文の修正に有用であると考えられる．

3. 文書添削システムの概要

本研究で開発している文書添削システムの基本的な構成，動作等について述べる．まず，本システムの構成を図 2 に示す．本システムは Web アプリケーションとして実装しており，システムで管理する資料や編集情報の保持，クライアント間での通信の中継等を行うサーバと，資料の閲覧，情報付加，資料のアップロードや削除等の操作を行い操作情報をサーバに送信するクライアントで構成される．以下で資料の閲覧・編集操作，資料の管理手法，複数クライアントでの同期について述べる．また，本システムでは HTML5 のオフライン Web 技術を用いてデータのキャッシュ等を行っており，それについても述べる．

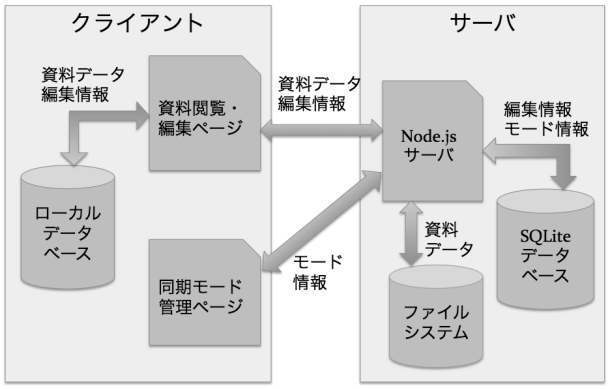

図 2: システムの構成

3.1 資料の閲覧・添削機構

本システムの資料の閲覧・添削機構について述べる．本システムでは PDF ファイルを主な添削対象として実装しており，PDF.js を用いて canvas 要素に PDF の描画を行っている．また，本システムでは PDF.js の機能を用いて，PDF ファイル内の文字列検索等も行う事が可能である．これらによって，基本的な閲覧操作を行えるようになっている．

次に添削機構について述べる．本システムでは，予めシステムで用意したインタフェースを用いて資料にアノテーションを付加する事で添削を行う．アノテーションとしては任意の文字列を保持するメモアノテーション，直線や楕円等の簡単な図形情報を保持する図形アノテーション等を実装している．またこのようなアノテーションの管理方法として，資料を描画した canvas 要素の上に同じ大きさの canvas 要素を重ね，その重ねた canvas 要素に対してアノテーションを描画している．これは，canvas 要素がラスタ画像としての情報しか持たないためである．資料とアノテーションを同じ canvas に描画した場合，アノテーションの追加は問題なく動作するが，アノテーションの変更や削除を行った場合に資料の再描画が必要となり，システムの処理負荷が大きくなる．資料を描画する canvas とは別にアノテーションを描画する canvas を用意することで，資料の再描画を行う必要が無くなり，システムの負荷を軽減できる．本システムにおける閲覧・添削の動作画面を図 3 に示す．

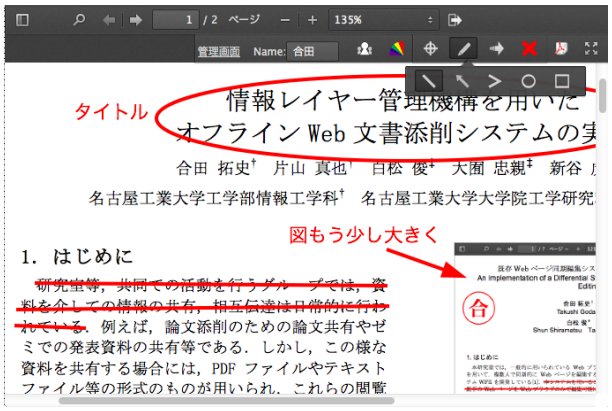

図 3: システムの資料閲覧・添削画面

3.2 資料の管理

資料の管理機構について述べる．本システムでは，図 4 に示すような資料管理用のミニウィンドウを用意し，資料のアップロードや削除，サーバに保管された資料の一覧，表示する資料の選択が可能となっている．資料のアップロードはウィンドウ上部のドロップエリアに資料をドラッグアンドドロップすることで可能である．ドロップエリアの下に資料を分類するためのタブと，タブに分類された資料のサムネイルでの一覧が可能となっている．また，本システムではサムネイルでの資料一覧の他にリストでの資料一覧画面も実装している．リストによる資料一覧はサムネイルを用いた一覧と比較し一覧性が高く，多くの資料を表示する場合でも処理の負荷が小さい．そのため，サーバに保管されている全ての資料の一覧はリストにのみ表示し，そこからユーザが必要な資料をサムネイルによる一覧画面に移動させるという実装を行っている．本システムでは，資料の管理構造としては階層構造を取っており，タブによる分類の上位階層としてワークスペースによる分類も行っている．

3.3 複数クライアントでの同期

本システムは，複数人での協調添削も想定しており，閲覧状況やアノテーションのリアルタイムな同期機能を実装している．アノテーションの付加状況をリアルタイムに同期することにより，複数人で同時に添削を行っている場合でも即座に他人の添削を知る事ができるため，重複した添削が起き難くなる等のメリットが期待できる．また，本システムでは添削操作だけではなく，スクロールの位置やズーム倍率等の閲覧状況も同期する事が可能となっている．これによって，被添削者が添削者の添削状況をリアルタイムに見る事が可能となっており，遠隔地で直接指示を出しながらの添削が可能となっている．ここ

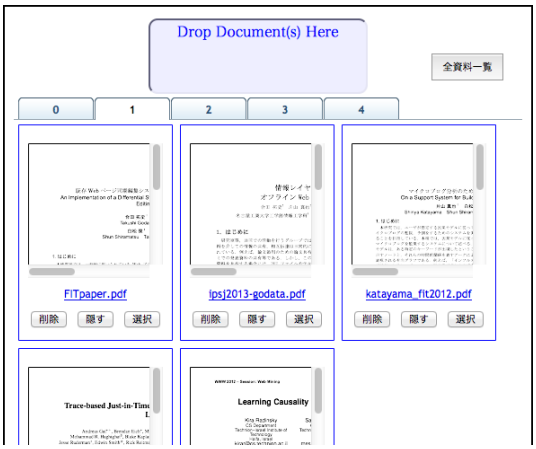

図 4: 資料管理インタフェース

で，添削状況に関しては常に同期されても問題ないが，閲覧状況に関しては常に同期した場合，添削者が複数人いる場合に不都合が発生する．そのため，閲覧状況の同期に関しては同期モードの管理画面を用意し，操作できるようになっている．システムでは閲覧状況の送信を行う添削モード，閲覧状況の受信を行う閲覧モード，閲覧状況の同期を行わない自由モードを実装している．

3.4 オフライン Web 技術を用いたデータの管理

本システムでは，HTML5 のローカルデータベースを用いて資料データ，アノテーションデータ，サムネイルのレンダリング後の画像データ，ユーザ設定データの保存を実装している．HTML5 のローカルデータベースとしては，Web Storage，Web SQL Database，Indexed Database API の 3 つが利用可能であるが，これらはブラウザによって実装状況が異なる．本システムは PDF の描画を PDF.js に依存しており，PDF.js の動作環境は Safari と Firefox となっている．そのため，本システムの動作環境もそれに準ずる．それぞれのブラウザで利用可能なローカルデータベースとしては，Safari が Web SQL Database と Web Storage，Firefox が Indexed Database API と Web Storage となっている．本システムでは，ユーザデータの格納にはブラウザによらず Web Storage を利用して格納している．Web Storage は利用できる容量が 1 オリジンあたり 5MB と資料のデータ等を保存するには少ないが，利用のための実装が単純であり，また多くのブラウザで利用可能である．そのため，大きな容量を必要とせず，利用方法が単純であるユーザデータの保存に Web Storage を用いて実装している．資料データ，アノテーションデータ，サムネイルのデータに関しては，ブラウザに応じて Indexed Database API もしくは Web SQL Database を利用する．資料のデータ等は比較的大きな容量を必要とし，Web Storage による管理が困難であるためである．

4. 論文添削支援のためのアプローチ

4.1 差分検出機構

論文添削において，修正前後の論文の差分を検出するためのアプローチについて述べる．修正による差分をシステムで示すことができれば，修正箇所の確認が円滑に行え，添削者の負担軽減に繋がると考えられる．差分検出のためのアプローチとして，まず考えられるのが被添削者による明示である．本研究の論文添削モデルにおける提出プロセスにおいて，被添削者が添削プロセスで与えられた指示に応じて修正した部分を，アノテーションを用いる等の手段で修正後の論文に示すことで，添削者はどの部分が変更されたかをすぐに知る事ができる．このアプローチは特に実装を必要とせず，被添削者の手間によって行う事が可能である．しかし，被添削者の負担となることや，明示漏れの可能性があるため，システムが自動で支援を行うことが望ましい．システムが自動で行うためのアプローチとしては，PDF の文字列を抽出し比較する方法が考えられる．まず，比較を行う 2 つの PDF ファイルを A，B とする．サーバにこれらの PDF がアップロードされた段階でサーバで文字列抽出を行い保存しておく．そして，これらの PDF の差分を抽出する場合は，2 つの PDF から抽出した文字列情報を比較することによって文字列の差分情報を取得する．そして，比較によって得られる文字列を適当な単位に分割する．後は PDF.js の機能を用いて PDF 内の文字列検索を行い，検索結果をハイライトによって示す．ここで，文字列比較によって得られる差分文字列の分割単位ををどのようにするかは検討する必要があるが，論文内に同一の文が複数回登場することは少ないと仮定し，文単位での分割を行う予定である．

4.2 バージョン管理機構

論文の添削に限らず，複数人で 1 つのシステムや文書に対して編集を繰り返し行うような作業において，バージョン管理は有用である．バージョン管理システムとしては，ソフトウェア開発におけるバージョン管理を行う git が有名である．あるソフトウェアにおいて変更部分を追跡できるようにすることで，不具合の発見・改善や，何故そのような変更を行ったのかという意思の疎通等を効率良く行うことができる．このバージョン管理の考え方を論文添削に対して適用することで，論文に対してどのような変更が行われたかを追跡することができ，論文の執筆過程の知識を蓄積した資料として役に立つのではないかと考えられる．

まず，論文の添削作業におけるバージョンのモデルを図 5 に示す．本稿では，論文の添削作業におけるバージョン管理として，最初に添削を行った論文から，添削と修正を繰り返し最終的に完成した論文までの一連の論文の集合をプロジェクトと呼ぶ．また，プロジェクト内である時点でのスナップショットとなるものをリビジョンと呼ぶ．図の四角で示したものが被添削者が送信した論文であり，そこから伸びる矢印の先にある楕円で示したものが添削者による添削情報である．ここで，” 修正案 0-A” とは，リビジョン 0（プロジェクトにおいて最初に送信される論文の初期稿）に対して添削者 A が行った添削を示す．また，図にあるように，同一のリビジョンに対して別の添削者 B が添削を行う場合も考えられ，それを” 修正案 0-B” として示している．そして，それら 2 つの添削情報から伸びる矢印の先にあるものが，リビジョン 0 に対して” 修正案 0-A”，” 修正案 0-B” を適用したリビジョン 1 となる．本研究では，複数人で論文の執筆を行った場合でも各リビジョンにおいて各々の修正の統合を行うものと考え，リビジョンは逐次的に増加すると仮定している．このように添削を繰り返し，適当なリビジョンにおいて論文が完成し，プロジェクトは終了する．本研究ではこのようなバージョンのモデルを想定している．

本システムでは，システムの資料管理機構であるワークスペースやタブ，それに加えてアップロードするファイル名を用いて，ユーザによるバージョン管理が可能である．例えば，あるプロジェクトに対してワークスペースを作成し，タブを用いてリビジョンを管理する等である．しかし，この方法によるバージョン管理はワークスペースやタブの作成等ユーザの負担

が大きく，ワークスペースやタブが多く作成されることになるため，それらの資料分類機構としての利用が困難になるという問題がある．そのため，この機構とは別にバージョン管理を行う機構を実現することが望ましい．

本研究においては，論文のファイル名を用いてのプロジェクト分類を試作する．論文の添削ではあるファイルに修正を重ねて行われる場合が多いため，ファイル名が添削サイクルの途中で変更される事は少ないと考えられる．従って，ファイル名をプロジェクトの識別子として有効なのではないかと考えられたためである．具体的な実現について述べる．現在，サーバでの PDF ファイルの保管としては，サーバの資料保存ディレクトリ以下にワークスペース名，タブ名の順にディレクトリを作成し，そこに PDF ファイルを保存している．これをワークスペース名，タブ名，プロジェクト名の順のディレクトリ構造とし，PDF ファイル名にリビジョン番号を用いる事で管理する．ファイルのアップロードが行われた場合は，ファイル名から同一のプロジェクトがあるかどうかを判定し，あればその中に新しいリビジョンとして PDF を保存，無ければ別のプロジェクトとしてディレクトリを作成し PDF をリビジョン 0 として保存する．過去のリビジョンの閲覧としては，サムネイルによる資料一覧画面の対応する資料の部分に対して，リビジョンを移動するインタフェースを追加することで実装できる．本システムにおける添削情報となるアノテーションのバージョン管理については，どのリビジョンに対するアノテーションであるかを属性値としてリビジョン番号を追加することで実現できる．

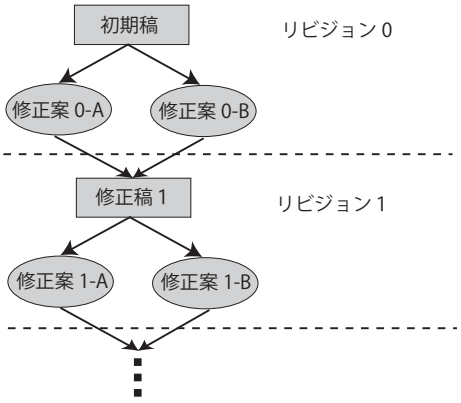

図 5: 論文添削におけるバージョン管理モデル

4.3 プロセス終了通知

添削者・被添削者の間で，円滑に添削・修正の終了を伝えるための機構の実現について述べる．この支援については，次の２ステップが考えられる．ステップ１は添削・修正の終了検知であり，ステップ２は具体的な通知手段である．

まずステップ１の添削・修正の終了検知について述べる．本研究において想定している論文添削サイクルにおいては，添削の終了は添削プロセスの終了，修正の終了は提出プロセスの終了として考えられる．各プロセスの終了判定として，添削プロセスはシステム内に添削終了を通知するためのボタン等を配置し，添削が終わった際にユーザが操作するという手段により行う．この場合でもユーザが通知のための操作を行う必要はあるが，判定はほぼ正確に行う事ができ手間も十分に少ないと考えられる．他に添削者のシステムからの切断等によって自動で添削プロセスの終了を検知する方法も考えられるが，誤検出等の可能性を考えるとメリットよりもデメリットの方が大きいのではないかと考えられる．提出プロセスの終了については，単純に論文のアップロードによって検知することができる．

次にステップ２の具体的な通知手段について述べる．現在通知の手段として，以下の３つが考えられる．まず１つ目に，システム内でポップアップ等により通知する機構が考えられる．この手段では高いリアルタイム性を得られるが，添削者や被添削者が常にシステムにアクセスしている必要があり，主な通知手段として有効であるとは言い難い．しかし実現の容易さやリアルタイム性の高さ等のメリットから他の手段と併用するには有効であると考えられる．２つ目の手段としてはメールの配信が考えられる．プロジェクトにおける添削者，被添削者の情報を予めシステムに登録しておき，通知が必要な段階で登録された添削者，被添削者に対して自動でメールを送信する．この場合，システムにアクセスしていなくとも通知を行うことができるが，頻繁に添削を行う場合はシステムから多くのメールを受信する事になり，ユーザに不便を強いることが予想される．また，システムでユーザのメールアドレスを管理する機構も必要となる．３つ目の手段としては Twitter 等の SNS を利用した通知が考えられる．この場合メールを用いた通知と違いシステムを利用するユーザのメールアドレス等をシステムで管理する必要が無く，システム内での通知のようにシステムに常にアクセスしておく必要もない．問題としては，システムからの通知に気づかない可能性がメールに比べて高い事が考えられる．本研究では，ツイッターを用いた通知機構を試作する．

4.4 添削内容の競合提示

論文添削における添削内容の競合解消におけるアプローチについて述べる．競合としてはある添削内容に関して同様の指示を行う場合，無関係な指示を行う場合，矛盾する指示を行う場合の３つが考えられる．これらについて添削の内容に応じて競合の解消ができれば理想的であるが，実現するのは高度な自然言語処理が必要となり困難である．そのため，本研究ではこれらの競合を検出し，提示することで，被添削者が競合する添削内容の比較・検証を行いやすくし，競合解消を支援することを目標とする．本研究においては，付加されたアノテーションの種類に応じて，アノテーション座標を用いた競合の検出を行う．本システムではアノテーションを用いて添削を行うが，そのアノテーションが保持する位置情報を用いて，距離がある閾値以下となる場合を競合と判定して示すことで実現する．

5. おわりに

本稿では，論文の添削システムの実現を目的として実装を行っているシステムについて述べ，それにおいて具体的な添削支援を行うための手法について提案を行った．具体的な添削支援の手段として，モデルの提案等を行いながら差分検出，バージョン管理，添削のプロセス終了通知，添削内容の競合提示について具体的な実現方法について述べた．提案した手法の内，差分検出の機構とバージョン管理については，論文添削における有用性は高いと考えられ，優先して実装を行う予定である．

参考文献

[合田 13] 合田拓史，片山真也，白松俊，大囿忠親，新谷虎松，”情報レイヤー管理機構を用いたオフライン Web 文書添削システムの実現” 第 75 回情報処理学会全国大会論文集 vol.75 no.4 pp.241-242 Mar. 2013