ネットワーク利用者が持つ情報の集約化支援機構とその評価

東京工業大学 大学院
情報理工学研究科 数理・計算科学専攻
門間 信行
(学籍番号 96M37279)

1997年度修士論文

指導教官 大野 浩之 講師

1998年1月30日
(1998年3月2日改訂)

目 次

第1章

はじめに

情報の集約化とは、多くの情報の中から有用なものを選別し、1つにまとめ上げる行為である。広域ネットワークにおいては、ある質問とそれに対する回答をまとめた『FAQ集』の作成が、この代表例である。このように、ネットワーク利用者が提供する情報を集約することによって、その情報としての価値を大きく向上させることができる。しかし、情報の集約作業は、高度で知的な作業であるため、編者に大きな負担をかける。この負担を軽減するための試みとして、複数の人々による分担作業によって情報の集約化作業を行なう事例がある。集約された情報の蓄積には、WWW(World Wide Web)の利用が適しているが、広域ネットワーク上に分散した人々がWWWを介して情報の集約化作業を行なう枠組はない。これは、WWWの協同作業の場としての機能の不足が主な原因である。

本研究では、情報の集約化作業にかかる編者の負担を軽減することを主目的とし、協同作業による情報の集約をWWWを介して実現する手法を提案する。集約作業における作業の各過程を計算機の支援によって自動化し、人にかかる負担の軽減をはかる。

また、WWWに不足している協同作業の場としての機能を補うために、集約作業によって生成される文書をハイパーテキスト構造で表し、これを保存・管理するための独自のハイパーテキストデータベースおよび、データベースを操作するためのハイパーテキストシステムを開発した。このハイパーテキストシステムは、ハイパーテキスト構造に基づくバージョン管理機構の導入、HTMLに依存しないハイパーテキスト構造の表現などの機能により、WWWに不足している協同作業の場としての機能を補う。また、WWWとのインターフェースとしてWWWの通信プロトコルであるHTTPに若干の拡張を加えた通信プロトコルを持っている。これをJava Servletを用いて実装を行なうことによって既存のWWWサーバのバックエンドとして動作するため、WWWの利用者であれば集約作業に参加することが可能であり、かつ誰もが集約作業の場を作成できる。同時に、集約作業を支援する上で有効である以下の特長を持っている。

- 複数の人の担当部分が複雑に入り混じる文書でも、各部分ごとの著者を明確にできる。
- ハイパーテキストの参照モデルであるDexterモデルに基づくデータモデルを用いることにより、文書の所有者以外でも文書の部分参照やリンクの生成、文書の組み合わせが可能である。

本論文では、このシステムの運用結果に基づく評価および機能面での評価を行ない、情報を集約する作業において本システムを用いることの有用性および編者にかかる負担が軽減されることを示した。

本論文の構成について説明する。第2章では、情報の集約化の背景と、WWWを作業の場とした協同執筆作業による情報の集約化について述べ、従来の協同執筆支援の研究やシステムの実例について紹介する。第3章では、協同執筆作業による情報の集約化において、作業を実現するためのモデルと、新たな情報サービスシステムの必要性について述べる。第4章では、第3章で示すモデルに基づき、集約作業がどのように進展するかについて説明する。第5章では、集約化支援のために開発したサービスシステムの設計と実装について述べる。第6章では、システムの設計や運用に基づく評価を行なう。第7章では、今後の課題について説明する。

第2章

背景

2.1 情報の集約化の意義

多くの情報の中から有用なものを選別し、1つにまとめ上げる行為である「集約」は、日常では頻繁に行なわれている。集約の例としては、新聞記事から特定のテーマに関する記事を抜き出してのスクラップ、編集担当者が記者から届けられる多くの原稿から必要な原稿を選び出しての誌面構成、論文の執筆における方々から寄せられた加筆やコメントなどの中から最適なものを選び出しての元の文書への反映、などが挙げられる。

集約化は、単独では乏しい価値しか持たない情報を、関連する情報と結合させることにより、その価値を高める手法であると言える。

このような行為は計算機を利用して広域ネットワーク上でも行なわれている。代表例として、広域ネットワークの利用者のための情報交換の場として利用されているUsenet Newsに投稿される記事の中から、頻繁に質問される事柄とそれに対する回答を抽出し、1つにまとめたFAQ集の作成事例や、WWWのような情報発信システムにおいて、自分の発信情報に対する意見を第三者から募り、それを発信情報に反映させるような事例が挙げられる。

集約によって得られた文書は、集約される以前の情報を提供した側と、集約作業を行なった側による協同作業の結果、生成された文書であると言える。集約作業を行う側を意識せずに情報提供側が一方的に情報提供を行なった場合、集約作業は大変な労力を伴う作業となる。この労力の軽減のためには、集約化を協同作業の一環として位置付け、情報提供側を協同作業の枠組に取り込む必要がある。

本論文では、このような協同作業、特に協同執筆による集約化に注目する。また、協同作業の場として広域ネットワーク上の情報サービスシステム、特にWWWに注目する。さらに、計算機支援によって集約化作業の負担の軽減をはかり、作業全体の効率化を実現するための手法について考察する。

2.2 協同執筆作業とその計算機支援

本節では、協同執筆作業を計算機支援するにあたり、まず協同執筆の作業形態について分類し、計算機支援によってサポートすべき作業形態について述べる。さらに、協同執筆

を計算機によって支援する既存のシステムの実例について述べる。

2.2.1　協同執筆の作業形態

Preece らによると、文書の協同執筆には、三つの形態がある [Pre94]。三つの形態とは、「執筆作業の分担」による形態、「レビューの繰り返し」による形態、及びこの「二つの形態の混合」の形態である。

執筆作業の分担 あらかじめ章などを単位として文書をいくつかの部分に分けておき、それらの執筆作業を協同作業者に分担させる方法である。この形態の場合、分割された文書のそれぞれの部分においては書き込みはその部分の担当者一人にのみ許可される。他の部分の担当者は書き込みに関する権限を持たない。

レビューの繰り返し 執筆者としての権限を一人に持たせる形態である。すなわち、執筆者が作成した原案に対して、協同作業者はコメントづけなどの校正を行なう。それを基に、執筆者は原案を書き直す。この形態の場合、文書に対しての書き込みは執筆者一人にのみ許可され、他の協同作業者にはコメント付けのみが許される。

二つの形態の混合 上記した二つの形態が混合して行われる場合である。

実際には一つの文書を作成するときでも、三番目の形態、すなわち、二つの形態が連続的に繰り返し現れる場合が多いとされる。

近年、協同執筆作業に対して計算機を使って支援を行なうことで、作業全体の効率を上げるためのシステムが多数開発されている。したがって、このようなシステムでは、協同執筆作業の形態のうちどちらか一方をサポートするだけでは不充分であり、両方をサポートする必要があると言える。

2.2.2　従来の協同執筆支援システム

本小節では、従来の協同執筆支援システムについて述べる。具体例を挙げる前に、対象とする範囲を定めておく。

グループウェアとしての協同執筆支援システムの位置づけ

協同執筆作業を計算機によって支援するシステムは、グループでの作業を支援するグループウェアとして位置づけられる。グループウェアは、支援の対象とする作業の特性によっていくつかに分類される。

第一の分類として、時間的特性があり、複数のユーザが音声や画像を共用ウインドウシステムなどを介して同時に作業を行なうリアルタイム型、および電子メールや電子掲示版のような蓄積型の通信を基本とした蓄積・非同期型に分けられる。第二の分類として、空間的特性があり、複数のユーザが会議室などに集合して対面で使用する同室対面型、およ

び地理的に分散した複数ユーザがリモート通信機能を用いて使用する遠隔分散型に分けられる。石井 [石井 94a] によれば、協同執筆を支援するシステムは、このうち、蓄積・非同期型および遠隔分散型に分類される。

現在では、グループメンバーがリアルタイムで文書を吟味しながら執筆作業を進める形態を支援する、すなわちリアルタイム型の協同執筆支援システムも存在するが、以下では、蓄積・非同期型および遠隔分散型に属する協同執筆支援システムについて紹介する。

初期の協同執筆作業支援

協同執筆を支援する初期のシステムでは、小規模のグループの構成員が非同期に執筆作業を行ない、かつ完成した文書はグループの総意であることを前提としていた。また、文書に対して直接に書き込むための機能よりも、レビューとして文書にコメント付けをする機能に着目している。これは、前述した「レビューの繰り返し」による執筆形態のサポートに重点を置いた結果である [石井 94b]。

非同期型の協同執筆作業を支援するためのシステムには、Broderbund Software 社の ForComment や Quilt[LFK88]、PREP[NKCM90]、Mainstay 社の MarkUp などがある。これらのシステムは、電子メールのような通信手段を用いて文書を構成員間で回覧しつつ、構造化電子メールやハイパーテキストの技法を用いて文書に対するコメント付けを行なう機能を提供している。文書とコメントの構造関係の管理のためにハイパーテキストの機能を利用する試みは古くから行なわれている。Quilt や PREP は文書および文書に付加されたコメントをハイパーテキストを用いて管理し、ユーザに提示するシステムである。ハイパーテキストに関しては、次章で詳しく述べる。

広域ネットワーク上での協同執筆作業支援

広域ネットワークの普及により、広域に分散した作業者の間で協同執筆作業を行なう事例が生じた。これにともない、広域ネットワーク上での協同執筆をサポートするためのシステムが開発されている。

前述の ForComment などのシステムは、主に部署内などのローカルエリアネットワーク上での利用を想定している。コメント付けのための通信手段として電子メールを用いている点などは広域ネットワーク上においても適用できると思われるが、基本的には広域ネットワーク上での利用は考慮されていない。

協同編集作業を広域ネットワークを用いて行なうにあたっての問題点は Pacull らによって指摘されている [PSS94]。広域ネットワークでの協同編集作業では、文書が保存される共有空間と、各作業者のローカルな作業空間の間のコネクションを常に維持することが現実的でないために、システムによる各作業者の状況の把握は困難である。そのため、複数の作業者が同時に編集作業を行なう事態が生じた場合、各作業者の作業空間内にある編集途中の文書の整合性を保てずにある作業者の編集結果が無効になることがある。これは、”lost update problem” と呼ばれ、協同執筆作業において解決すべき重要な問題である。

この点を考慮したシステムとして、DUPLEX[PSS94] が提案されている。DUPLEX で

は、上記の問題の対策として、文書を章などの独立した部分に分割し、作業者が分割の単位を自由に変更できる機能を備えることで、複数の作業者による同時編集の可能性の低下させている。

2.2.3　WWW の協同作業環境としての利用

近年、WWW を人々の協同作業の場として利用しようとする試みが盛んに行なわれている。WWW は広く一般に普及しており、多くの利用者による協力が期待できる点や、情報の蓄積、保存が容易である点などがその理由であるが、特に協同執筆の点においては、WWW のハイパーテキスト機能が注目されている。WWW は、文書の共有のために各人が協調しながらそれぞれの視点にもとづいてハイパーテキストにおけるリンクを張れるために、分散作業環境を実現するために有益であるとされる。

また、WWW の広域性に着目して、従来のグループウェアの研究で対象とされていた小規模なグループの枠組みにとらわれず、利用者が不特定多数の場合でも適用可能な協同作業支援システムが多数開発されている。

WWW を用いて協同執筆を行なうためのシステムには、WWW 上に存在する文書に対してコメント付けを行なう機能を備えたアノテーションシステムがあり、Commenter[RWP95] が代表的である。Commenter では、任意の Web ページの任意の箇所に、誰もがコメント付けを行なえ、これらのコメントを多くの利用者間で共有することにより、WWW を協同作業の場として提供している。また、WWW を 議論を行なう場として利用するために、WWW のページに対するコメント付けの方法として Usenet News のような記事を投稿できる仕組みを持たせたシステムも多数開発されており [D$^+$95]、HyperNews[LB95] や CoNote[DH] がその代表である。これらはアノテーションシステムの一部であるが、特に Web Conferencing System と呼ばれ区別されている。

しかし、これらのシステムの多くは多人数での文書のレビューを目的とし、WWW 上に存在する文書に対するコメント付けに焦点が当てられており、執筆作業の分担をサポートするシステムは少ない。これは、WWW が協同作業の場、特に協同執筆作業の場として備えている機能が非常に限られている点が原因と考えられる。特にハイパーテキストシステムとしての機能には制限が多い。この点について次小節で述べる。

2.2.4　ハイパーテキストとその関連研究

ハイパーテキストは、関連する文書の速やかな参照を行うための技術であり、Nelson や Engelbart らによって本格的に研究され始めた分野である。ハイパーテキストでは、情報は「ノード」と呼ばれる小さな単位に分割される。「アンカー」はノードの部分を示す。アンカー同士は「リンク」によって結びつけられており、利用者はノードからノードへリンクをたどることによって情報の中を航行できる。

ハイパーテキストの概念には、ある作家の作品の大部分を利用しつつ、第三者が新たに独自の作品を出版するといった例が示されており [Nel87]、協同作業において有効な仕組みとされる。前述のように、協同執筆作業を支援するシステムである Quilt や PREP は文書

および文書に付加されたコメントをハイパーテキストを用いて管理し、ユーザに提示するシステムである。

NoteCard や HyperCard のような初期のシステムでは、ハイパーテキストの構成要素であるノードやリンクは単一の計算機内に展開されるにとどまっていた。しかし、近年の広域ネットワークの発達にともない、ネットワーク全体を巨大な分散データベースと見立てるような、分散ハイパーテキストシステムが実現可能となった。分散ハイパーテキストシステムでは、ハイパーテキストのノードを広域ネットワーク上に展開してその間を、広域ネットワークを経由するリンクで結ぶことになる。

ハイパーテキストシステムとしての **WWW** とその問題点

中でも World Wide Web は単純な仕組みによって分散ハイパーテキストを実現している[BLCGP92]。前述したように、WWW は広範に普及し、現在では最も広く利用されている広域情報システムとなっている。後述する本研究において開発したシステムでも、情報提供の手段の一つとして WWW を利用する。これは、情報システムとしての WWW の特長である、情報を蓄積可能である点や即時性の点に着目した結果である。

しかし、WWW は個人やグループの情報発信システムとして確立されているものの、協同作業の場として利用するには不十分な仕組みとなっている。これは、WWW を構成する諸要素のうち、ハイパーテキストのノードやリンクを記述するための標準形式として採用されている HTML[BLC95] の表現力が低い点、及び広域ネットワーク間でハイパーテキストを転送するためのプロトコルである HTTP[BLFF95][FGM$^+$97] が状態を持たない (stateless) プロトコルである点に起因している。

HTML では、ハイパーテキストのリンクはノードの中に埋め込まれている。この方法は単純であり、取り扱いが簡単であるが、ノードに対してリンクを生成できるのは文書の所有者のみに限られてしまう。また、WWW ではリンクを表す標準形式が HTML 以外には存在しないため、文書にリンクを付加するためには、文書の所有者でさえ、その文書を HTML に変換しなければならない。すなわち、HTML で書かれていない文書に対してリンクを表現する標準の方法は存在しない。

また、WWW では、ネットワーク経由で編集作業を行う枠組はほとんど提供されていない。HTTP は ハイパーテキスト構造に関しては HTML で記述されることを前提として設計されたプロトコルである。従って、自分が所有者である Web ページを編集するときでさえ、HTTP を用いてハイパーテキスト構造を変化させる作業、すなわち、アンカーの付加やリンクの生成などは行なえない。

一方、ハイパーテキストの研究では、複数人でノードやリンクを共有する形の協同作業について考慮したモデルの考案やシステムの開発が行われてきた。その一例である Dexter 参照モデル [HS94] や Device Hypermedia[GHMS94][GT94] では、ノードとリンクを分離し、別々に保持する方式を採っている。この方式では、ノードに対する書き込み権限がなくても、そのノードに対するアンカーやリンクの付加が可能である。この方式を WWW に適用する試みも行われている。Microcosm モデルの DLS[CRHH95] や ComMentor[RWP95] がその代表例である。

IETF の WEBDAV(World Wide Web Distrubuted Authoring and Versioning) ワーキンググループ (以下、WEBDAV-WG) の活動は、Web ページの内容をサーバ以外のクライアント上で変更することを目的としており、これには協同執筆作業の支援も含まれている。サーバ上での Web ページのデータのバージョン化など、WWW に不足している協同執筆作業の場としての機能を補うために、WWW の仕組みそのものを拡張する多数の提案がされている。しかし、HTML の使用を前提としているため、拡張される機能にはバージョン管理された文書の表現力などの点で限界が生じている。

2.3 まとめ

本研究の目的は、WWW を作業の場とした協同執筆による情報の集約化を計算機支援して集約化作業における編者の負担を軽減し、集約化作業の効率化を実現することである。

作業形態としては、蓄積・非同期型および遠隔分散型を取り扱う。また、レビューのための文書に対するコメント付けではなく、執筆作業を分担するために文書を直接書き直すことができる機能のサポートを目標とする。

WWW を協同作業の場として利用するにあたっては、ネットワーク経由で編集作業を行う枠組の欠如、および WWW のハイパーテキストシステムとしての機能の欠如が問題となる。

次章では、これらの点を解決するためのシステムの構築にあたり、対象とする協同作業のモデル化、システムがハイパーテキストシステムとして満たすべき要件について述べる。

第3章

協同執筆による集約化方法

3.1 協同執筆の定義と対象

本研究の目的は、WWW を作業の場とした協同執筆による情報の集約化を計算機支援によって集約化作業における編者の負担を軽減し、集約化作業の効率化を実現することである。

情報交換の参加者が文書の書き手となり、集約化された文書の作成に協力できるような作業の枠組を提案し、かつ計算機によって途中の各作業過程を支援して集約化作業を行なう人の労力を軽減することにより目標の達成をはかる。作業形態としては、蓄積・非同期型および遠隔分散型を取り扱う。

このようにして作成された文書および作成途中の文書は複数の人が書いた部分が入り混じった文書となる。本稿では、このような文書を「協同文書」と呼び、協同文書を執筆する行為を「協同執筆」と呼ぶ。

協同執筆のような協同作業に関する従来の研究は、社内グループなどの比較的小さな、かつ結びつきが強く、お互いが信頼できるグループを対象にしていることが多い。一方、本稿で対象とするのは、コンピュータネットワークの上での比較的大きな規模の、互いが顔見知りでもないような弱い結びつきのグループをも含む。

また、前章で述べた協同執筆の形態のうち、「レビューの繰り返し」はアノテーションシステムによってサポートされる。しかし、アノテーションシステムでは、既存の文書を編集することはできないため、「執筆作業の分担」はサポートされない。本研究では「執筆作業の分担」のサポートに注目する。コメント付けに関しては、既存のアノテーションシステムとの連携によってサポートすれば良いという立場を取る。

3.2 ハイパーテキストを利用した協同文書の表現と保存方法

本節では、本研究で採用する協同文書の表現方法や保存方法について説明する。

表 3.1 引用を行なっているメッセージの例

```
> ○○○をインストールしたのですが、
> うまく動きません。
> どこが悪いのでしょうか。
○○○はあなたの環境では
うまく動作しないといわれています。
私には原因は良くわからないので、
もっと詳しい人に聞いて下さい。

> 他に似たようなソフトウェアはありませんか。
◎◎◎や×××がおすすめです。
動作が軽くて、設定も簡単です。
```

3.2.1 「窓つき文書」による協同文書の表現

他の人が新たに書き足すことによって文書を作成する行為である「引用」を例にとり、協同文書の表現方法について述べる。

電子メールやUsenet Newsに流れるメッセージに対する回答メッセージは、元のメッセージから適度に文脈を引用をすることにより、そのメッセージ単体でも、つまり元のメッセージを捜し出して読み返さなくても、意味がわかるように工夫されている場合が多い。引用を行なっているメッセージの例を表3.1に示す。

数週間に到着したあるニュースグループの記事を調べたところ、明確に回答である記事(レスポンス記事)のうち、元記事を全く引用していない記事は3%程度に過ぎず、引用の行為は広く定着していると言える。

他の文書の一部を引用している文書は、図3.1のような、向こう側が見えるように「窓」がつけられた文書というモデルでとらえられる。この、「窓つき文書」を用いて引用を表現した様子を図3.2に示す。「窓と窓の対応づけ」によって、窓の向こう側を見ることができる。

「窓つき文書」の中には引用元の文書は一切含まれていないので「窓つき文書」の著者は一人であるといえる。「窓つき文書」を単位として、協同文書における個人の権利や責任の境界とできる。また、引用だけでなく、一般の校正行為、すなわち加筆、削除、訂正、複写、順序入れ換えなどは全てこの「窓つき文書」で表現できる[門間97a]。

したがって、協同文書を表す構造として、「窓つき文書」を用いることは適当であるといえる。また、協同文書とは、「窓つき文書」をたくさん重ねて、上から見たものである。

以上で述べた「窓つき文書」をハイパーテキストの用語で説明すると、「文書部分」がノード、「窓」がアンカー、「窓と窓の対応づけ」がリンクに相当する。

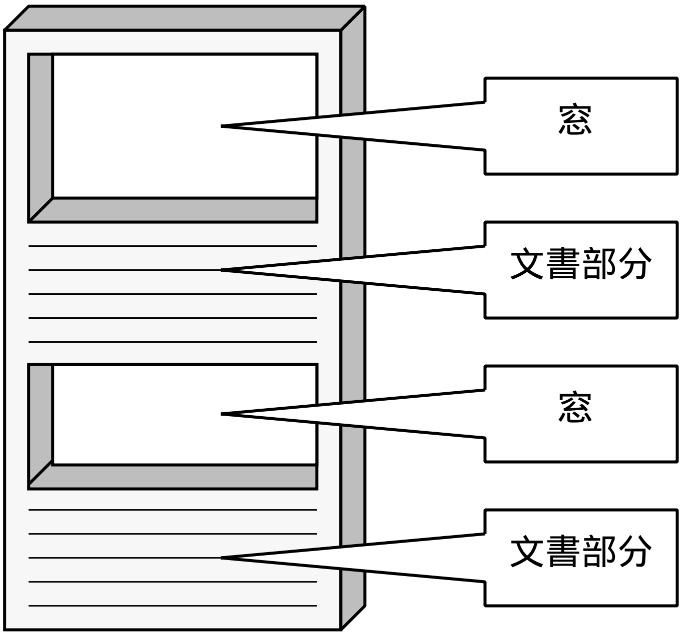


図 3.1 「窓つき文書」

3.2.2 協同文書の保存先

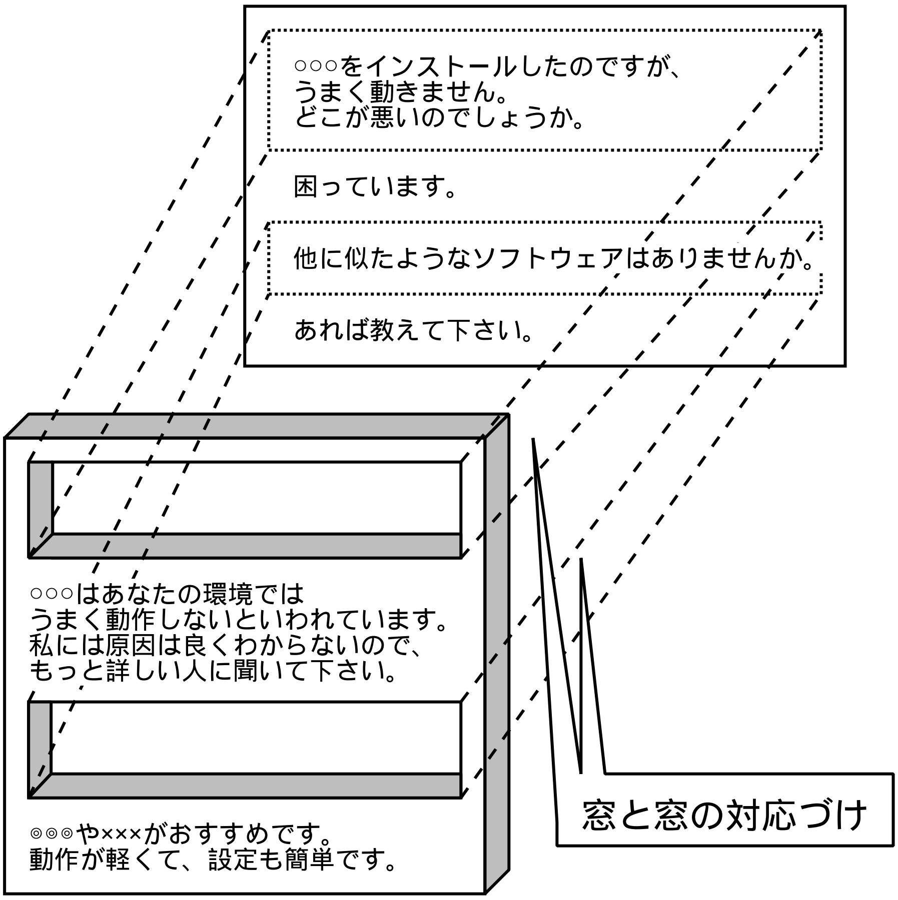
協同文書を表す構造として窓つき文書を用いた場合、すなわち、元の文書に付加された「窓」、新たに作成された文書部分、それに付加されている「窓」、さらに「窓と窓との対応づけ」の4つの新たなデータが生成される。ここでは、これらを計算機ネットワーク上のどの計算機に保存するかについて述べる。

集中型 協同文書を表す構造を全て、協同文書を管理する単一の計算機の元に保存する方法である。

この場合、協同文書を管理するサーバを1つ立ち上げれば良く、他の計算機の資源を長期に渡って必要としない。また、協同文書の構成要素は単一の計算機内に一括管理されるので、窓つき文書の「窓」の位置が変化した場合などにおいて、整合成の保持が容易である。

一方、協同文書の著者は必ず自分の執筆部分を協同文書の管理者宛に送付しなければならない。このため、電子署名などの機能を用いて、管理者の手元での文書の改ざんなどの行為に対処する必要がある。また、協同文書を管理する計算機には、協同文書を保存するために多くの記憶領域を用意しておく必要も生じる。

分散型 窓つき文書の「文書部分」を、新たに文書を生成したユーザのローカルな広域情報システム内(自分のWebページなど)に保存する方法である。この方法の利点として、文書の著者が自分の編集結果を常に手元に置くことができる、文書を保存するための記憶容量をほとんど気にしなくて良い、などの利点がある。



○○○をインストールしたのですが、
うまく動きません。
どこが悪いのでしょうか。

困っています。

他に似たようなソフトウェアはありませんか。

あれば教えて下さい。

○○○はあなたの環境では
うまく動作しないといわれています。
私には原因は良くわからないので、
もっと詳しい人に聞いて下さい。

◎◎◎や×××がおすすめです。
動作が軽くて、設定も簡単です。

図 3.2 「窓つき文書」を用いた引用の表現

しかし、n回重ねられた窓つき文書を参照する場合には、協同文書は最大n台の計算機に分かれて保存されるので、n台の計算機全てにアクセスしなければ協同文書を完全に取り出すことができない。ネットワークの現状から、これはユーザにとって大きなストレスとなることが予想される。

また、全てのユーザが広域情報システム内に自由に使用できる情報空間を所有しているわけではないため、作業に参加することができない人々が生じてしまう。

以上の理由から、ある文書に対する窓は全て、元の文書と同一の管理下に置かれるものとして、話を進める。

しかし、上で挙げた分散型で問題とされる点は徐々に解決されつつあり、また、既存のWWWのページやUsenet Newsの記事など、膨大な量の既存の広域情報ネットワーク資源を協同文書の一部として利用可能にするためには、分散型のほうが都合が良い。しがたって、分散型への移行をも考慮に入れる。この場合、協同文書の書換えにともなう、分散した「窓つき文書」間の整合性を保つためのサーバ間による通信、アクセス速度向上のための協同文書のキャッシュなどについて考慮する必要がある。

3.3 作業モデルの提案

システム構築にあたり、まず作業モデルを導入する。

作業モデルでは、前節を踏まえ、「窓つき文書」が協同作業者によって作られてから、協同文書の管理者の元に送られて保存されるまでの過程を示す。

以下では、作業の基本方針について述べ、次に具体的な作業モデルを提案を行なう。

3.3.1 基本方針

情報を集積し、協同文書の作成を行なう人を「主催者」と呼ぶ。協同文書は、主催者によって所有され、管理される。主催者は、自分のWWWのような広域情報サービスのページを情報交換のための場として用意し、あらかじめUsenet Newsなどを通じて協同文書作成への参加を広く呼びかけておく。他の人々は協同文書を直接ではなく、主催者に対しての働きかけを起こすことによって、間接に操作を行なう。以下、他の人々を「情報提供者」と呼ぶ。情報提供者から寄せられた文書は、主催者のもとでデータベース内に保存され、広域情報サービス上で公開される。

3.3.2 作業モデル

協同執筆作業のモデルとして、図3.3に示すような作業モデルを導入した。

情報提供者はWWWを通じて協同文書を入手し、それに変更を加える。変更結果を元の協同文書が存在する広域情報サービス内のデータベースに登録しておき、その上で、変更の承認を得るために主催者に変更点の内容を送付する。主催者は送付された内容を見て、承認/非承認を判断し、承認された場合は文書の差替を行なう。この一連の作業を繰り返すことにより、協同文書は次々と更新される。差替は自動的に行なわれるので、文書の更新にともなう主催者の労力は軽く済む。

以下で各過程について述べる。

差分作成 情報提供者は、適当なエディタなどを用いて文書に変更を加え、新しい版の文書を作成する。変更内容は、変更前後の文書の相違、すなわち差分の形で保存する。この差分が「窓つき文書」である。

差分登録 情報提供者は、差分を主催者側のデータベースに登録する。変更後の文書は、差分と変更前の文書から復元することができる。ただし、この時点では文書の差替は起こらず、変更があったことが示されるのみである。

なお、主催者はこの過程を拒否することができる。このとき、差分のデータベースへの登録は行なわれないままの状態で、次の過程へ進む。

連絡 次に、情報提供者は作成した差分を主催者へ送付する。連絡手段としては、確実に主催者の手元に届くサービスを用いることが求められ、電子メールを用いることが最も適切であると考えられる。

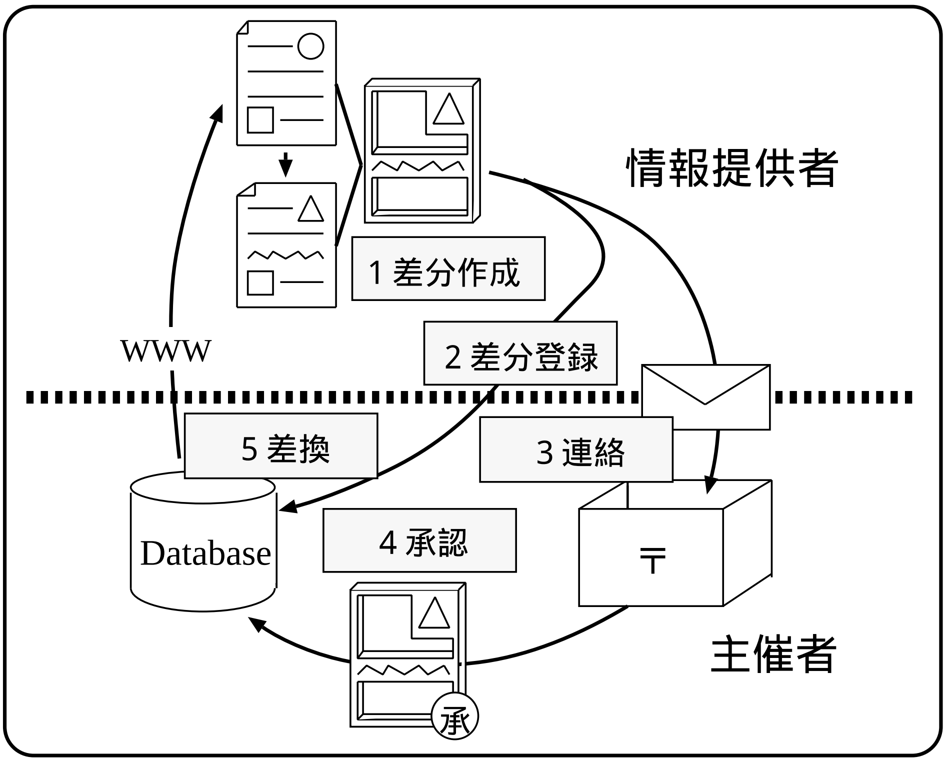


図 3.3 作業モデル

承認	主催者は受け取った差分を見て、この変更を受け入れるかどうかを判断する。この過程は、お互いを信頼できるグループ内でのみ作業が行なわれる場合などにおいては省略することができる。
差替	変更が承認された場合、差分登録過程においてデータベースに登録された変更後の文書をこれを新しい版とし、文書を更新する。差分登録過程が拒否されていた場合は、この部分で差分を受け取り、差分と変更前の文書から変更後の文書を復元し、これを新しい版として文書を更新する。

3.3.3 2 つの登録方法

主催者は、差分登録過程を拒否することができる。このため、作業の流れとして、2 通りのパターンが考えられる。

差分登録過程を経ない場合

- 差分作成
- 連絡

- 承認
- 差替 (この過程で差分の登録が行なわれる)

差分登録過程を経る場合

- 差分作成
- 差分登録 (この過程で差分の登録が行なわれる)
- 連絡
- 承認
- 差替

前者の場合、承認前の文書がデータベースへ登録されて公開されることを防ぐことに相当し、いたずらの書き込みを防ぐ効果がある。しかし、この場合、データベースへの登録は承認過程の後になるため、書き込みの内容がすぐには掲載されない。後者はその逆となる。

筆者が以前に示した作業モデル [門間 96][門間 97a] には、差分登録過程が存在せず、作業は全て前者の流れで行なわれていた。この場合、承認作業を頻繁に行なわない限り、同じような内容の登録および承認要求が何件も寄せられることになり、承認過程における主催者の負担は高くなる。このため、差分登録過程を設けることにより、この問題の解決をはかった。しかし、上に述べたような両者の長短の関係から、どちらを採用するかは主催者の裁量とした。

3.4 新たな情報サービスシステム

「窓つき文書」の構造を保存して WWW を用いて公開し、かつ協同執筆を可能にするため、新たに情報空間を用意してその上に新たなサービスシステムを構築することを考える。

協同作業の場として WWW に不足している機能を補い、協同文書を柔軟に操作、管理する機能を提供することにより、協同執筆による集約化の結果である協同文書の生成を導くと共に、協同文書のみならず、既存の文書資源の再利用性を高めるシステムを目指す。新たなサービスシステムには WWW とのインターフェースを持たせることにより、WWW 上での協同執筆作業環境を構築する。

「窓つき文書」の保存・管理を行なうためのハイパーテキスト機能としては、以下のような要件が求められる。これらの要件は、従来の WWW の仕様では満たされていない。

- ネットワーク経由で文書の操作が可能であること。
- 文書の一部分を取り出して参照できること。

  「窓つき文書」の「窓」を表すには、既存の文書のうちの任意の一部分を取り出すことができなくてはならない。

- 文書の部分同士を組み合わせて、新たな文書とすることができること。

  「窓つき文書」は、複数の「窓」部分と「文書部分」の組み合せから成る。したがって、「窓」に相当する既存の文書の一部分や、新たに書かれた文書の部分などを自由に組み合わせて、新たな文書とすることができなくてはならない。

- 文書の部分ごとの著者を取得できること。

  「窓つき文書」では、単一の文書内に複数の人によって書かれた部分が混在する。したがって、読者が必要に応じて部分ごとの著者を取得できる必要がある。

第4章

作業モデルに基づく協同文書の更新経過

前章で述べた作業モデルに従うと、実際の更新の過程はどのように進むかについて述べる。その上で、承認過程における主催者にかかる負担は、情報提供者や主催者が取る行動や、協同文書の質、時間とどのように関係するのかについて考察する。

4.1 作業モデルの待ち行列化

一般に、情報提供者の数が複数人であるのに対して主催者は一人である。主催者側の過程の処理能力が、情報提供者からの連絡の到着頻度に比べて劣っている場合、次々と到着する承認要求に対処し切れず、結果として、寄せられた貴重な情報提供を捨てざるを得ない状況に陥る。

以下では、情報提供者側から寄せられるどの程度の頻度の承認要求に対して、主催者側にどの程度の処理能力が必要となるのかの見積もりを与える。

作業モデルにおける主催者側の過程、すなわち承認過程と差替過程に注目する。情報提供者からの承認要求を客、主催者が承認や差替にかかる時間をサービス時間とみなし、サービスを受ける窓口は1つ、待ち行列長の最大値が1であるような単純な待ち行列モデルを考えることができる。待ち行列に関する諸要因を以下のように与える。

情報提供者からの承認要求の到着間隔 X は、ランダムな到着、すなわち指数分布に従うとみなす。一方、サービス時間 Y は承認過程のサービス時間と差替過程のサービス時間に分けられる。承認過程は、到着した文書を読み、受け入れるかどうかを判断する過程である。したがって、承認過程のサービス時間は、到着した文書の文字数や複雑さ、および、人間が文書を読む能力に依存するが、ここでは文書の文字数や複雑さはまとめて扱い、文書の文字数で代表させることとする。承認過程のサービス時間 $Y1$ を、以下のように定式化する。

$$\text{承認過程のサービス時間}\,Y1 = \text{到着した文書の文字数}/\text{単位時間あたりに読める文字数}$$

同様に、差替過程のサービス時間 $Y2$ を、以下のように定式化する。

$$\text{差替過程のサービス時間}\,Y2 = \text{書換える変更の文字数}/\text{単位時間あたりに入力できる文字数}$$

$Y = Y1 + Y2$ となる。

ここで、元の文書の文字数 $= l$ とおく。また、簡単にために、

$$到着した文書の文字数 = 元の文書の文字数 = l、$$

$$書換える変更の文字数 = P \times 到着した文書の文字数 = Pl(0 < P < 1)、$$

とすると、$Y1$、$Y2$ はそれぞれ、

$$Y1 = l/単位時間あたりに文書を読む文字数$$

$$Y2 = lP/単位時間あたりに入力できる文字数$$

となる。

Y の値を定量的に見積もるために、人間が単位時間あたりに文書を読む文字数を 500 字/分、人間が単位時間あたりに入力できる文字数を 50 字/分とすると、１回の承認にかかる時間 $Y = Y1 + Y2$ は以下のようになる。

$$Y1 + Y2 = l/500 + Pl/50(分)$$

さらに $E(P) = 0.1$ とおくと、

$$\begin{aligned} E(Y1 + Y2) &= Y1 + E(P)Y2 \\ &= l/500 + l/500 \\ &= l/250(分) \end{aligned}$$

となる。$l = 1000$ 字 なら、$E(Y)$ は 4 分、$l = 10000$ 字なら、 $E(Y)$ は 40 分 である。l $= 10000$ 字の場合、主催者が 24 時間承認過程および差替過程の処理を行な ったとしても、処理できる能力は 1 日平均 36 件が限界である。

計算機支援による処理能力の向上

以上では人間が処理する場合を考えたが、承認過程および差替過程を計算機支援することにより、処理能力を向上させることは可能である。以下では、計算機支援によ って、どの程度主催者の処理能力を向上させることができるについて、おおまかな見積もりを得る。

差替過程は、特に主催者の判断の余地がないため、計算機支援によ ってほぼ完全に自動化できる。一方、承認過程は、主催者が文章を読み、判断を行なう過程であるから、完全な自動化は不可能である。しかし、文章全体を読まなくても判断が下せるように支援を行なうことは可能である。主催者は、文書の全部でなく、変更部分のみを取り出して読むことで承認を行なうことができるとすれば、承認過程のサービス時間 $Y1$ および差替過程のサービス時間 $Y2$ は、以下のようになる。

$$Y1 = Pl/人間が単位時間あたりに読める文字数$$

$$Y2 = Pl/\text{計算機が単位時間あたりに入力できる文字数}$$

ここで、$Y2 << Y1$ とみなせば、$Y = Y1 + Y2 = Y1$ となり、 Y は $Y1$ のみで表される。この場合の Y の平均 $E(Y)$ を求めるために、その他の記号の値をこれまでと同様に代入すると、

$$\begin{aligned} E(Y) &= E(Y1) \\ &= E(P)l/\text{人間が単位時間あたりに読める文字数} \\ &= E(P)l/500 \\ &= l/5000(\text{分}) \end{aligned}$$

となる。これは、計算機支援を行なわない場合の20倍の性能である。$l = 10000$ 字の場合、主催者が24時間承認過程および差替過程の処理を行なうと、処理できる限界は1日平均620件となる。この数値に対応できる人数は、ネットワーク利用者全体の人数と比較すれば、非常に限られた集団である。しかし、ネットニュース記事に寄せられる回答記事の量を考えると、この程度の処理能力でも実用に耐えるものと思われる。

4.2 承認が逐次行なわれる場合のバージョン推移

作業モデルに基づくと、協同文書に対して変更が行なわれた場合は必ず、主催者による承認が行なわれてから差し替えられる。

承認が一件の変更のたびに逐次、行なわれた場合、文書の更新は最もスムースに進むといえる。なぜなら、主催者は、現在と変更後の2つの版とを比較し、変更後の版が受け入られられるものと判断すれば承認すれば良く、そうでなければ却下すれば良いからである。この場合、比較の対象となる版の数は常に2しかない。この場合のバージョンの推移は図4.1のように直線的に推移する。

しかしながら、変更一件ごとに逐次に承認を行なうためには、一件の変更があった場合、承認が行なわれるまでの間、新たな変更を認めないような制約を設けること、もしくは、主催者が常に承認要求があったことを監視し続け、承認要求があったときに素早く対処することが必要となる。

前者のように制約を設けた場合、差分作成支援過程から差替過程に至るまでの間の時間、新たな変更が不可能となるため、作業が滞る。これは、作業に参加する意志のある人数が極端に少ない場合や、集約作業を終えるまでの期限が特に設けられていない場合にはそれほど問題とはならないとも思われるが、ネットワーク利用者が自由に作業に参加できる観点からは好ましくない制約である。

一方、後者のように監視を続ける場合、主催者は、例えば利用している端末の前に長時間拘束されるような、常に承認要求のための準備を整えておくことが要求される。これは、主催者にとって大きな負担であり、集約における主催者の負担の軽減を目的とする本研究の主旨にそぐわないばかりか、非現実的である。

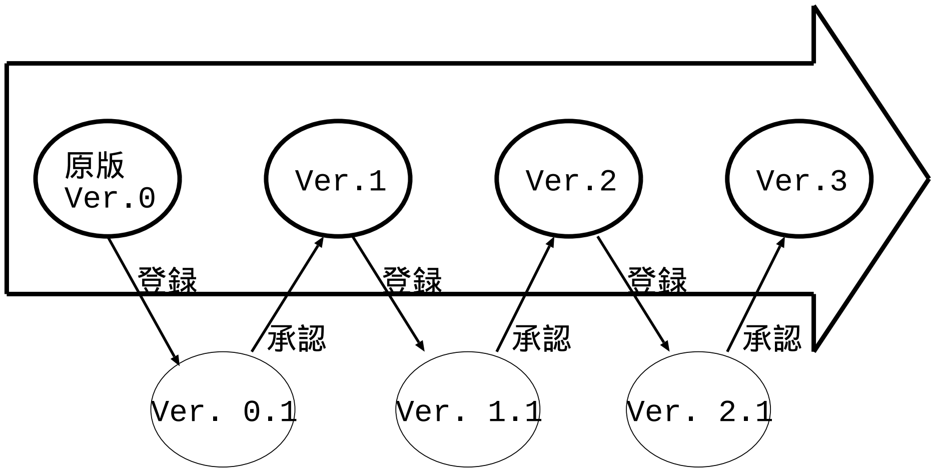


図 4.1 承認が逐次行なわれる場合のバージョン推移

4.3　承認がまとめて行なわれる場合のバージョン推移

作業モデルにおいて、変更一件ごとに承認を行なうのではなく、何件かの変更とそれに伴う承認要求をある程度まとめて承認する場合を想定する。承認を行なうタイミングは、承認要求に気がついたときに行なう、あるいは一定の時間ごとに行なうことが自然であると考えられる。

この場合、バージョンの推移は図 4.2 のようになる。バージョン n に変更を加えた結果、新たに作成された文書には、新しい順にバージョン n.1, n.2 ... と名づけている。

承認作業は、承認の対象となる複数個の分岐版の中から、次の版として最もふさわしい版を選び出すことに相当する。ただし、複数個の分岐版から最もふさわしい版を単純に選び出すことは難しい。個々の変更に対して、この変更はこの分岐版の内容ががふさわしいが、別の変更は別の分岐版の内容の方がふさわしい、などという場合が多いであろう。したがって、この承認作業における主催者の労力は、単純に分岐版の数に比例するのではなく、承認の対象となる分岐版の数が多ければ多いほど、比例関係よりも大きなオーダーとなることが予想される。また、承認過程の途中に新たな承認要求が届いた場合、この承認要求は前の版を書換えたものであるため、差替の候補とすることなく捨てざるを得ない。承認要求を捨てざるを得ないような行為はなるべく少なくすることが求められる。

4.4　承認がまとめて行なわれる場合の待ち行列

前節で述べたような承認がまとめて行なわれる場合について、どの程度の頻度で承認を行えば良いかを考察する。この場合の作業形態を以下のように待ち行列化する。

- 客の到着間隔 X は、ランダムな到着、すなわち指数分布に従うとみなす。

- サービスは、間隔 Z で行なわれる。このとき、待ち行列に溜っている客を一括に処

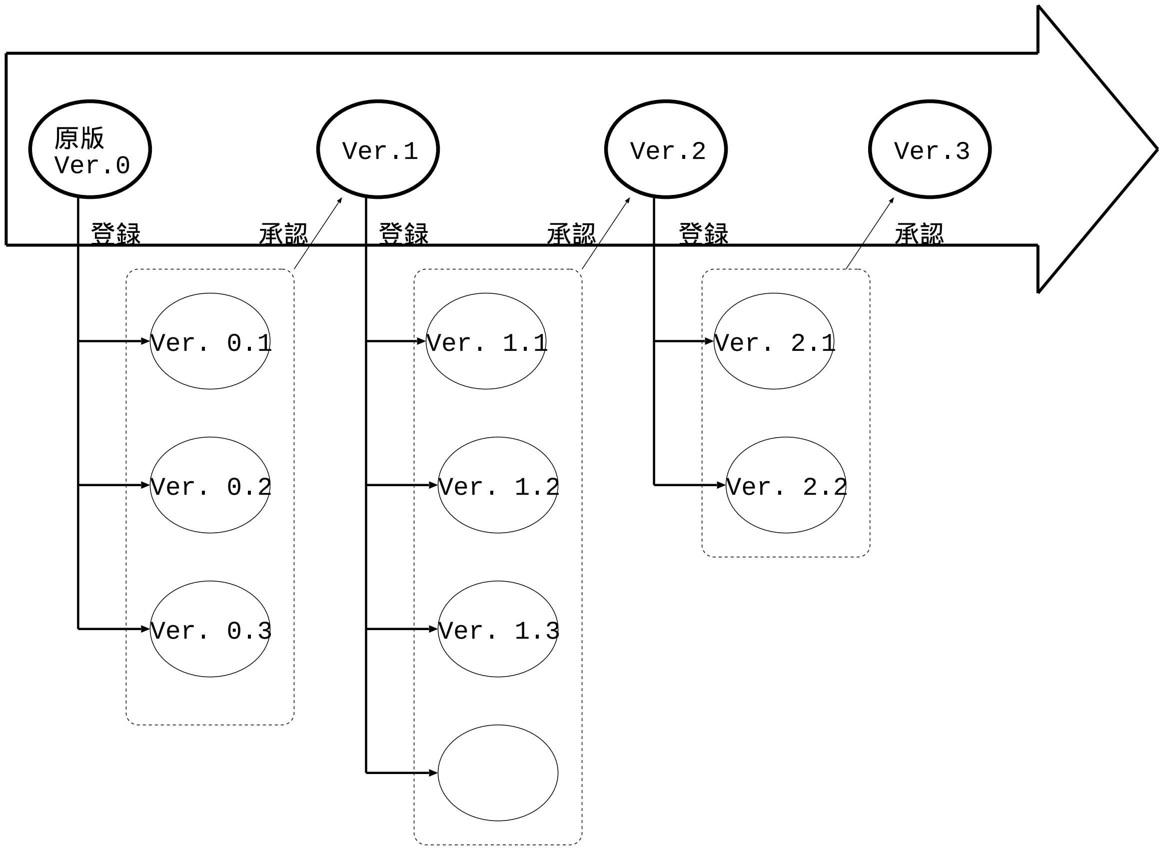


図 4.2 承認がまとめて行なわれる場合のバージョン推移

理する。

- 一括のサービス時間を Y とする。
- 他の客のサービス中に到着した客は、サービスを受けることができずに退出するものとする。

X はパラメータλ、平均 $1/\lambda$ の指数分布に従うものとする。Y は承認の頻度に対応するもので、$Y = 1/\mu$ の一定値 または パラメータ μ、平均 $1/\mu$ の指数分布に従うものとする。Z は、待ち行列に溜っている客の人数 N に依存するが、一人の客を処理するためにかかる時間を t として、ここでは $N^k t$ と見積もる。これらの条件のもと、全体の客数のうち、サービスを受けることができずに退出した客の割合、すなわち呼損率をシミュレーションによって求める。

シミュレーションにおける数値を、以下のように定める。t は前述の計算機支援を行なった場合の数値である $t = l/5000$ 分とし、$l = 10000$ 字とする。結果、$t = 2$ 分 となる。k は見積もりが難しいが、$k = 2$ とした。また、サービス間隔 Y の平均 $E(Y) = 1$ 日 とし、Y の分布が一様分布である場合と指数分布である場合それぞれについて、客の到着間隔のパ

ラメータ λ を変えてシミュレーションを行ない、呼損率が λ によってどのように変化するかを調べた。結果を図 4.3 に示す。

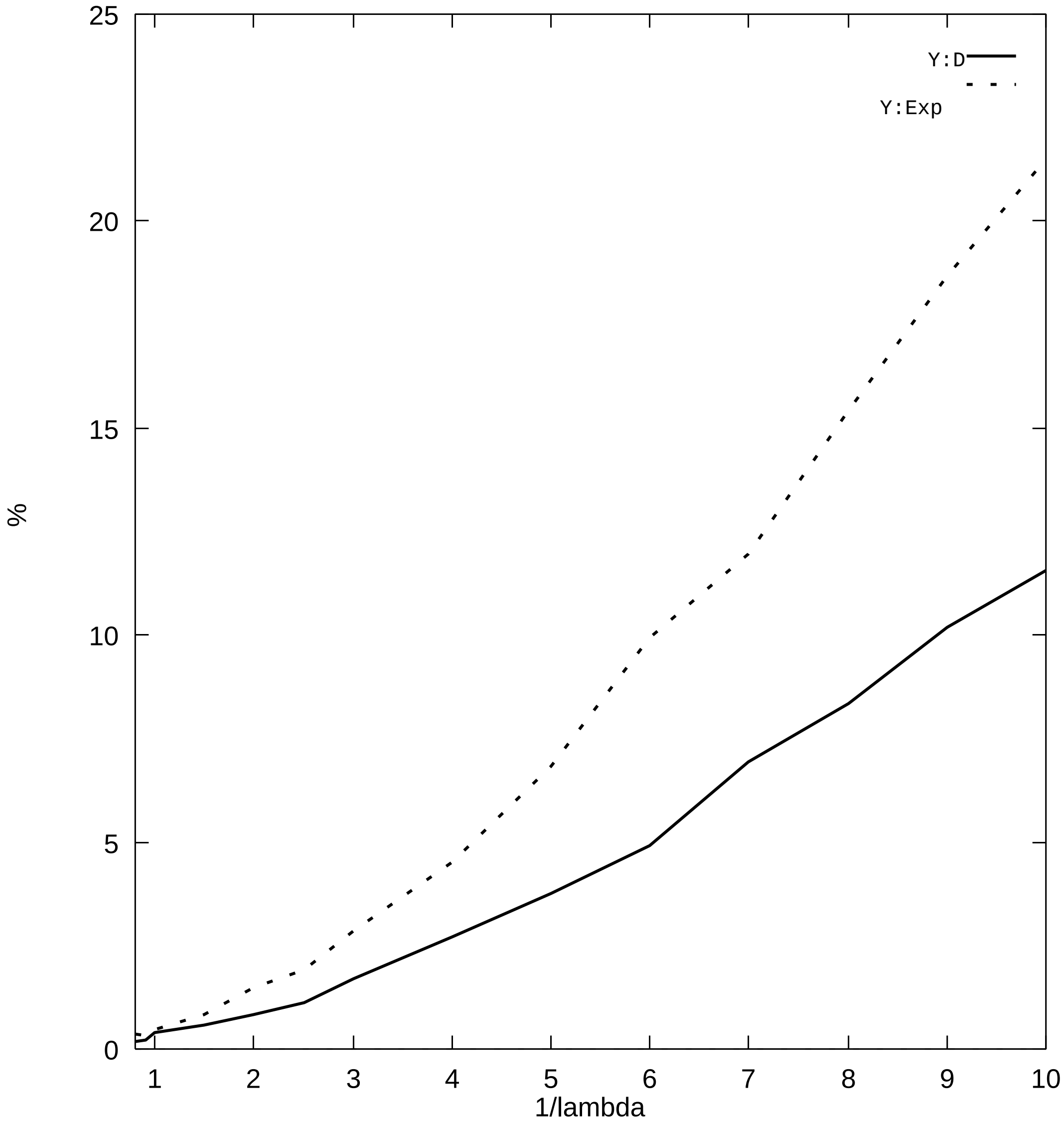


図 4.3 λと呼損率の関係

$E(Y)$ は同じでも、Y が一定値の場合のほうが、指数分布に従う場合に比べて呼損が半分程度少ない。これは、待ち行列の長さの平均が、Y が一定値の場合のほうが少ないためである。

また、両者とも $E(X) = 10$ 程度ですでに両者とも呼損率が 10%を超えており、やや実用に問題のある結果となった。しかし、これは k の値を 2 と大きく与えたためである。$k = 1$ とした場合は Y の分布には関係なく、$E(X) = 80$ 程度でようやく呼損率は 10%を超える程度である。k の値に関しては、実際にシステムを運用して調べる必要があるが、今回はそこまで至っていないため、今後の課題とする。

第5章

設計と実装

5.1　概要

前章までの事柄を踏まえ、ネットワーク利用者が持つ情報を集約するためのシステムの開発を行なった。本システムは、前述の作業モデルに基づいて文書の更新作業が行なわれることを想定しており、モデル中の各作業に計算機支援によって連続性を持たせることで作業における労力の軽減をはかる。特に、強く負担となっていた、文書の更新作業の自動化を実現する。

また、以下のような機能を提供する。

- 情報提供者がネットワーク経由で作業に参加できる環境の提供(World Wide Web の利用を想定)
- 主催者が目的に応じて協同文書の状態を制御する機構
- 読者への、協同文書の構造や著者の明示

5.2　システム構成

本システムの構成を以下に示す。本システムは、作業モデル中の5つの過程をそれぞれ支援するプログラムおよび、データベースを操作するプログラムの合計6つのプログラム、そして協同文書を保存するデータベースからなる。作業モデルと、その各過程を支援するプログラムの対応を図5.1に示す。

以下、6つのプログラムそれぞれについて説明する。

差分作成支援部　情報提供者による変更前後の文書の差分を、差換支援部が理解する形式で生成する。

差分登録支援部　生成された差分を解析し、文書の登録に必要な文書管理部が提供する操作の列を生成する。

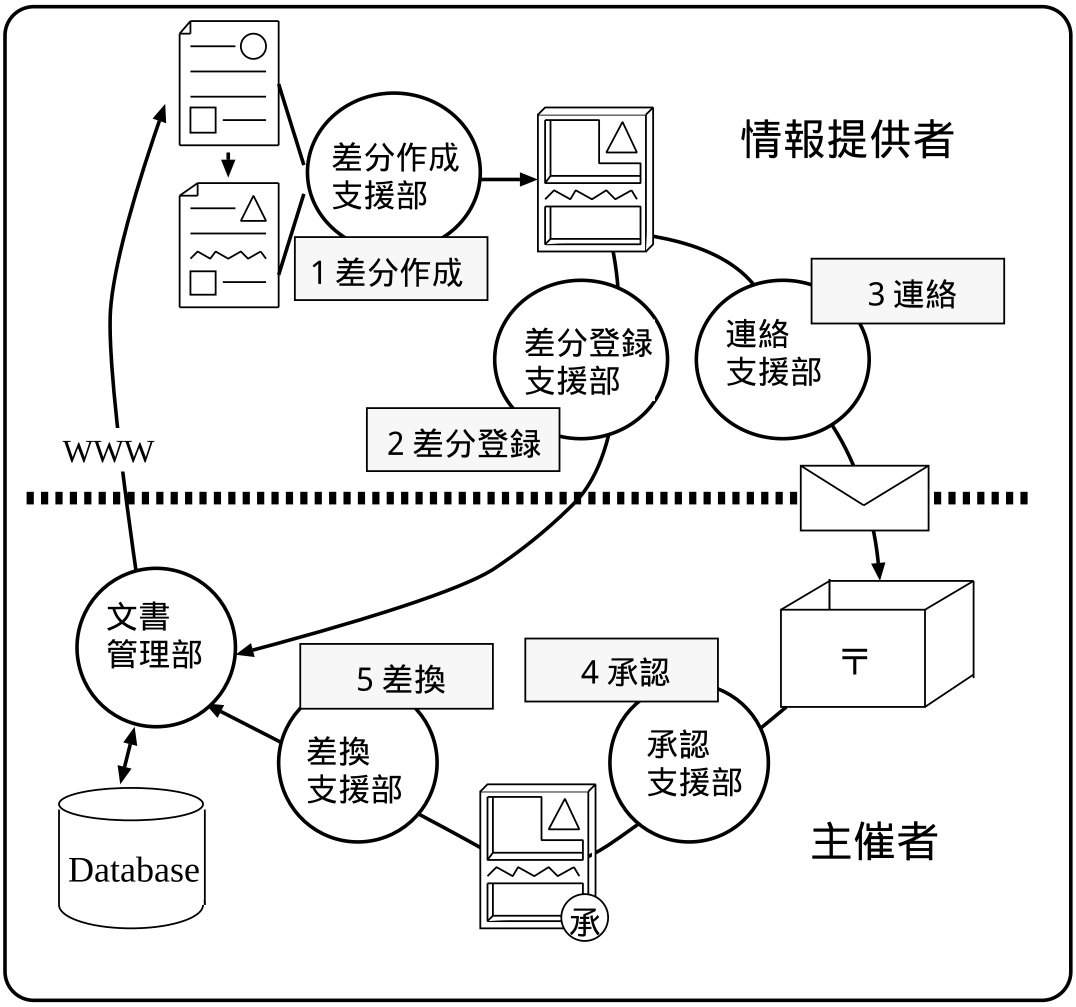


図 5.1 プログラム群

連絡支援部　情報提供者から主催者に対して変更の告知を行なう。

承認支援部　主催者による承認作業を支援する。差分は電子メールで送信されるので、メールリーダの一部として実現される。

差替支援部　送付された差分を解析し、文書の更新に必要な、文書管理部が提供する操作の列を生成する。

文書管理部　協同文書を管理するデータベース管理プログラムである。外部に対するインターフェースとして、データに対する操作を提供する。

ハイパーテキストデータベース　協同文書の構造は、「窓つき文書」を用いて表現できることは既に述べた。本システムでは、データベース中に「窓つき文書」を保存する。「窓つき文書」はハイパーテキスト構造である。つまり、文書部分がノード、窓がアン

カー、窓と窓との対応づけがリンクである。従って、本システムで構築するデータベースはハイパーテキストデータベースである。

文書管理部は、ハイパーテキストデータベースにおけるノードやリンクなどのハイパーテキスト情報を管理する [HS94]。たとえば、差分を元に差替が起こった場合、文書は実際に上書きされるのではなく、ハイパーテキスト構造を変化させることによって上書きされたように見せることになる。従って、新しい版とともに元の版はそのまま残されるので、過去の履歴を参照することが可能である。

5.3 実装

現在の実装は、編集の対象をテキスト文書のみに限定し、作業モデルにしたがって、最終的に情報をデータベース中に蓄積するものである。各部分に対応するプログラムの多くは、既存のソフトウェアに組み込まれる、あるいは既存のソフトウェアと連携しながら動作する。この様子を図 5.2 に示す。

以下の節では各部分について詳しく述べる。

5.4 差分作成支援部

変更前後の文書から、変更の内容、すなわち差分を、後で「窓つき文書」の形で表現できるように生成するための部分である。

変更前後の文書から「文書部分」と「窓」の部分、すなわち、各著者の新たに書いた部分とそうでない部分をいかに正確に取り出すことができるかは、この差分作成支援部によって生成される差分の質によって決まる。また、精度の良い差分を生成することにより、作業モデル中の「差分登録」や「連絡」の過程で生じるネットワークトラフィックを大きく削減することが期待できる。

したがって、差分作成支援部はシステム内で非常に重要な役割を果たす部分であると言える。

5.4.1 求められる要件

差分作成支援部において生成される差分は、変更前の文書と差分から変更後の文書を復元できる必要があることはもちろん、本システムにおいてはさらに以下の要件を満たすことが求められる。

細かい粒度での差分を取得できること 本システムでは、「窓つき文書」のモデルに従って、文書中の別々の人が書いた部分を別々に保存するため、差分の作成の際には純粋に元の文書に対して変更された部分だけを取り出す必要がある。すなわち、テキストデータを対象とすれば、差分は行単位のような大まかな単位でなく、文字単位のような細かい単位まで注目して生成される必要がある。

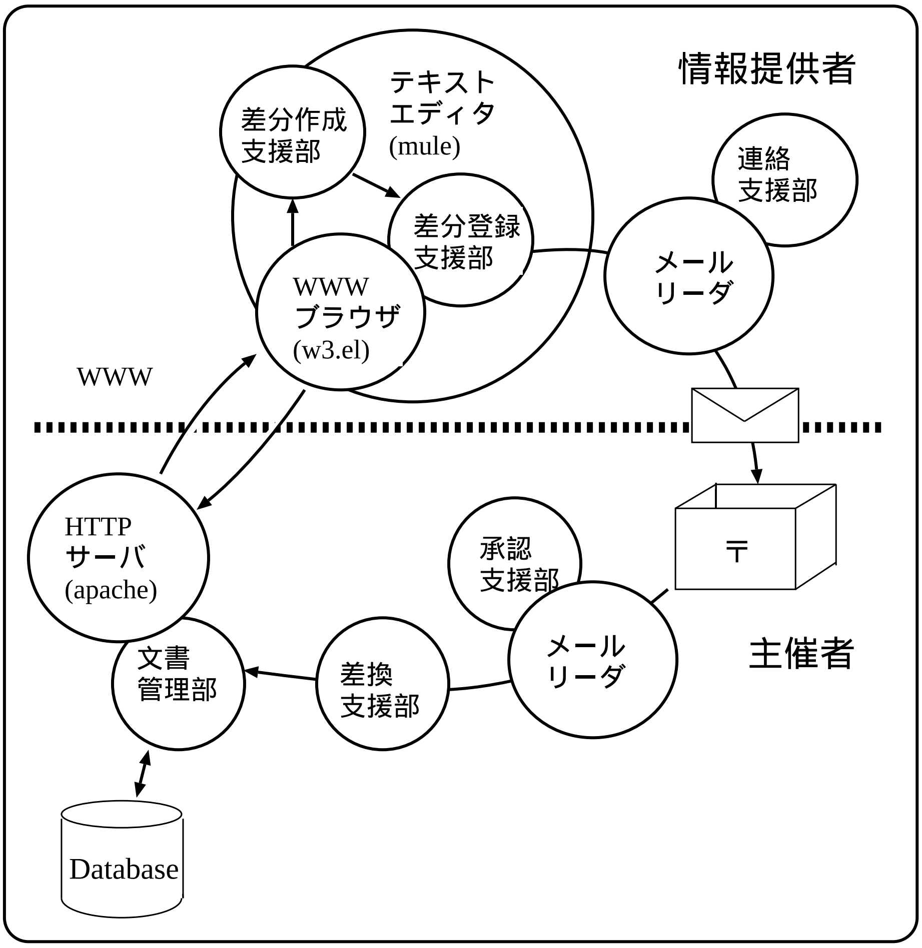


図 5.2 今回の実装と既存のソフトウェアの関係

多様な編集行為の情報を取得できること 同様に、文書に施された編集行為をの情報をなるべく正確に抽出する必要がある。すなわち、後述するように、「挿入」「削除」「上書き」の編集行為だけでなく、「入れ替え」や「複写」の編集行為についても検出できることが求められる。

本システムにおける精度の良い差分とは、以上の要件を満たす差分である。

diff コマンド

2 つのテキストデータの相違を得るための代表的な方法として、UNIX の diff コマンドを用いることが挙げられる。diff コマンドは、行単位で 2 つのテキストデータを比較し、新しく挿入された行、削除された行、内容が書き換えられている行、の 3 種類の変更を検知し、結果を出力するコマンドである。

しかし、diff コマンドではテキストを行単位で区切って比較を行なうため、得られる差分も行単位となる。したがって、細かい粒度での差分を取得することができず、上に挙げ

た1番目の要件を満たさない。

また、diff コマンドでは、差分を「挿入」「削除」「上書き」の3つの編集行為だけで表すため、上に挙げた2番目の要件も満たさない。たとえば、編集行為の一つである「入れ替え」は、あくまで「入れ替え」として取り扱うことが求められる。しかし、diff コマンドの解釈では、これを「挿入」および「削除」の2つの行為とみなしてしまう。これでは、情報提供者は新たに文書を「挿入」し、後で「削除」を行なったことになり、「挿入」された部分は情報提供者によって新たに書かれた部分との区別ができず、新たに書かれた部分とみなされてしまう。これは、実際の編集行為と合わない。

以上の理由から、diff コマンドのような比較的単純な方法では差分作成支援部の要求する機能を十分に満たすことができない。従って、要求する機能を十分に満たすための方法を検討する必要がある。

5.4.2 実現方法

差分作成支援部の実現方法としては、以下の2通りが考えられる。

文書の比較による方法 文書の編集が完了した後、編集前後の文書を比較することにより差分を生成する方法

エディタと連携する方法 エディタと連携し、エディタの機能である編集操作を追跡しながら差分を生成して行く方法

文書の比較による方法

編集の途中の過程でいかなる操作が行なわれたかに全く依存せず、文書の編集を完了した後で、編集前後の文章の差分を取る方法である。

この方法は、編集の過程が全くわからない文書同士の差分を取得したい場合でも適用でき、また、既存のさまざまなツールを用いることができる利点がある。しかし、上で挙げた diff コマンドに代表されるように、得られる差分は前述の要件を十分に満たすと言えない場合が多い。

また、この方法の一般的な特徴として、比較する2つの文書が似通っているほど、すなわち編集が行なわれた部分が全体に比べて少ない場合ほどうまく機能するが、そうでない場合は概して精度の悪い差分しか得られないことが多い点が挙げられる。

エディタと連携する方法

情報提供者は文書の編集の際にエディタを使用するが、エディタ内で行なわれる編集操作を追跡することによって、差分を作り上げる方法である。この方法では、差分の粒度を小さくすることができる。また、「入れ替え」や「複写」の編集行為はエディタで用意されている当該機能によって行なわれれば、検出が可能である。したがって、前述の要件を十分に満たす差分が得られる可能性が高い。

しかし、既存のエディタには編集操作の追跡がほとんど不可能なものもあり、この方法は全てのエディタに対して適用可能な方法ではない。また、過去のインターネットメッセージ内での引用のような、既成の文書に対しては、編集操作の履歴が残っていないのであるから、エディタによる編集操作の追跡はもちろん不可能である。

実現方法に関する考察

以上、2つの方法にはそれぞれ長短があり、精度の良い差分を得るには、どちらか一方の方法によらずに両者をうまく連携させる必要があると考えられる。「エディタと連携する方法」の適用が不可能な場合は、「文書の比較による方法」を取らざるを得ない。

5.4.3 実装

今回の差分作成部の実現方法としては、「エディタと連携する方法」を主に用い、広く一般に普及している文書エディタであるemacsに備わっている機能の中から、編集バッファ中のテキストの一文字ごとに属性値を付加することができる機能(property付加機能)を用い、エディタ内での基本的な編集操作を疑似的に追跡する。その結果に簡単なアルゴリズムを用いて解析を行なうことにより、精度の良い差分の生成を実現している。

具体的な方法を以下に示す。

1. emacsのバッファに編集するテキストを読み込む。
2. バッファ中のテキスト1文字ずつに対して、先頭からn文字目の文字にはnという属性値を与えておく[1]。
3. ユーザによる編集が行なわれる。
4. 編集終了後、文字の属性値に注目した解析を行なう。

編集後の文書中、編集前の文書から引続き現れる文字には属性値が与えられている。一方、新たに書き加えられた文字には属性値が与えられていないので、「挿入」の編集行為があったことがわかる。

また、emacsのproperty付加機能では、いわゆるカットアンドペーストを行なった場合でも、文字に与えられた属性値がそのまま保持される。このため、「複写」や「入れ換え」の編集行為を行なった場合でも、「挿入」の編集行為と区別することができる。

さらに、元の文書には、n文字目にはnという属性値を与えているため、編集前の段階では、文書中の各文字の属性値は前の文字の属性値よりも1大きく、次の文字の属性値よりも1小さいという規則が成り立つ。一方、編集後では、文書の属性値に注目すると、「削除」などの編集行為があった箇所では、この規則が成り立たない。したがって、この規則が成り立たない箇所で編集行為があったことがわかり、全ての編集行為を検出することができる。

---

[1] emacsのproperty機能では、デフォルトでは自己挿入(self-insert)された文字は前後の文字どちらかの属性値と同じになる(継承する)ように設定されているので、あらかじめこれを解除しておく

以上の解析を適用すると、編集後の文書のどの部分が元の文書のどの部分に対応しているのかを表す対応表を作成することができる。例として、表5.1のような文書を図5.3のような編集行為を通じて表5.2のような文書に書き直した場合に生成される対応表を図5.3に示す。表中の $(x\ y\ z)$ というエントリは、変更後の文書の x 文字目から y 文字目までの部分は、元の文書の z 文字目からと同じであることを示している。z が nil の場合は、元の文書に現れない、すなわち変更後の文書に新たに加筆された部分であることを示している。

表 5.1 編集前の文書の例

```
1 2 3 4 5 6 7 8 9101112131415161718192021222324252627
------------------------------------------------------
これは、元の文書です。
```

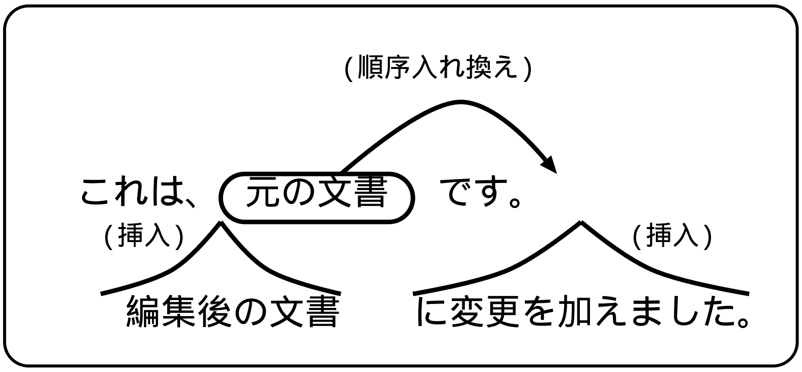


図 5.3 文書の編集の様子

表 5.2 編集後の文書の例

```
1 2 3 4 5 6 7 8 9101112131415161718192021222324252627
------------------------------------------------------
これは、編集後の文書です。元の文書に変更を加えました。
```

5.4.4 差分のフォーマット

以上の方法に従って生成された差分は、差分登録支援部や連絡支援部へ転送される。このとき、これらのプログラムが理解する共通の差分のフォーマットを取り決める必要がある。

表 5.3 変更前後の文書の差分を表す対応表の例

```
'((1 4 1) ; "これは、" に相当する部分
(5 10 nil) ; "編集後の文書" に相当
(11 13 9) ; "です。" に相当
(14 17 5) ; "元の文書" に相当
(18 27 nil)) ; " に変更を加えました。" に相当
```

表 5.4 差分対応表から GDIFF フォーマットへの変換の例

差分対応表	文字列	GDIFF フォーマットの解釈	GDIFF フォーマット
1 4 1		COPY 0 4	249, 0, 4
5 10 nil	”編集後の文書”	DATA ”編集後の文書”	6, ”編”,”集”,”後”,”の”,”文”,”書”
11 13 9		COPY 8 3	249, 10, 3
14 17 5		COPY 4 4	249, 4, 4
18 27 nil	”に変更を加えました。”	DATA ”に変更を加えました。”	10, ”に”,”変”,”更”,.(略)..”。”
		EOF	0

差分生成の際の要件で示した粒度の制約がない差分フォーマットとしては、文中に編集行為を表すタグを埋め込むによって差分を表現する VTML[VD95]、編集前の文書の部分をコピーする操作や、新規加筆部分を挿入する操作のような、復元操作の列を記述することによって差分を表す Generic Diff(GDIFF) フォーマット [vHP97] がある。VTML はテキスト (plaintext) 文書以外への適用が難しく、一方、GDIFF は復元操作をバイナリデータとして符号化するために人による可読性が低い。今回の実装では標準の差分形式として GDIFF フォーマットを用いているが、編集対象となる文書の種類によって適切なフォーマットを指定することが望ましいと考えられる。

表 5.3 で示した対応表から、GDIFF フォーマットへの変換は非常に簡単である。変換の例を表 5.4 に示す[2] 。なお、GDIFF フォーマットの解釈中の 「COPY x y」は、「元の文書の x 文字目から y 文字を、新しい文書にコピーする」という意味である。また、「DATA string」は、「文字列 string を新しい文書に書き込む」という意味である[3]

---

[2] GDIFF フォーマットでは、文書の先頭を 0 文字目と数える。一方、表 5.3 では先頭を 1 文字目と数えている

[3] 実際の GDIFF フォーマットでは日本語などの多バイト文字コードを扱うとき、ascii コードのような 1 バイト文字を扱うときとは少し違うコードを生成する。ここの例では説明のために、日本語文字を 1 バイト文字のごとく扱っている。

表 5.5 GDIFF フォーマットから拡張 HTTP への変換の例

GDIFF フォーマットの解釈	拡張 HTTP	拡張 HTTP の引数
COPY 0 4	ANCHOR old.txt	1 4
DATA ”編集後の文書”	POST old.txt	”編集後の文書”
COPY 8 3	ANCHOR old.txt	9 3
COPY 4 4	ANCHOR old.txt	5 4
DATA ”に変更を加えました。”	POST old.txt	”に変更を加えました。”
EOF	COMPOSE old.txt	(URL の列)

5.5 差分登録支援部

要件

差分作成部で作成された差分情報を受け取り、この情報を元に、文書管理部に対して編集行為を反映させるための操作を指示する。

差分作成部とは前節で述べた共通の差分フォーマットを理解することによって連携する。また、文書管理部とは共通のプロトコルを持つ必要があるが、後述する拡張された HTTP を使用する。

GDIFF フォーマットから拡張 HTTP への変換の例を表 5.5 に示す。

実装

実装の一例として Emacs Lisp を用いて実装を行ない、emacs で稼働する WWW ブラウザである w3.el に組み込んだ。

5.6 連絡支援部

要件と実装

連絡支援部では、差分作成部から受け取った差分を主催者宛に送出する。既存の電子メールの配送機構をそのまま用い、情報提供者が使用するメールリーダに組み込んで使用する。

実際には、受け取った差分をそのままメールリーダに引き渡すのみであるので、連絡支援部は前述の差分フォーマットを理解する必要もない。

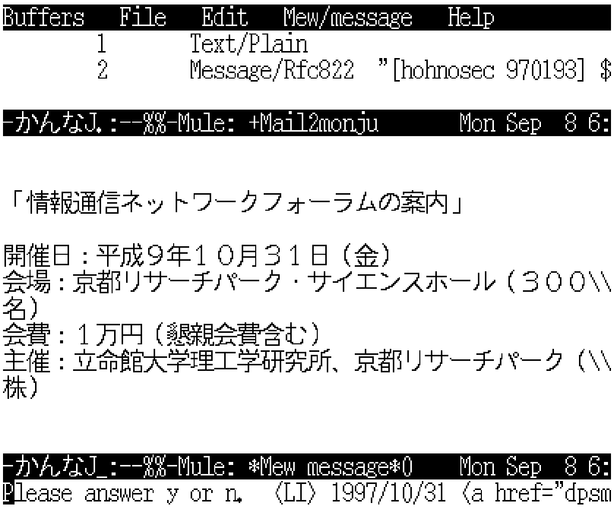
5.7　承認支援部

要件

情報提供者から送付された電子メールを受け取り、その電子メールの中から差分情報を取り出して主催者の承認を仰いだ上で、差替支援部に差分を送出する。

実装

実装の一例として、主催者が使用しているメールリーダが emacs 上で稼働している場合を想定し、Emacs Lisp によって実装を行なった。このプログラムは主催者の利用するメールリーダに組み込まれ、主催者は承認/非承認の決定を行なうことができる。承認された場合、連絡支援部から受け取った電子メールを差替支援部に送出する。承認の様子を図5.4に示す。図は三つの領域に分かれているが、図中の最上部がメールリーダの機能である電子メールの一覧を表示している領域、中部が電子メールの内容を表示している領域である。最下部はミニバッファと呼ばれる領域であり、図中ではミニバッファに承認を求めるメッセージが出力されている。ここで「y」を入力すると承認が行なわれる。



```
Buffers   File   Edit   Mew/message   Help
          1       Text/Plain
          2       Message/Rfc822  "[hohnosec 970193] $

-かんなJ.:--%%-Mule: +Mail2monju       Mon Sep  8 6:


「情報通信ネットワークフォーラムの案内」

開催日：平成９年１０月３１日（金）
会場：京都リサーチパーク・サイエンスホール（３００\\
名）
会費：１万円（懇親会費含む）
主催：立命館大学理工学研究所、京都リサーチパーク（\\
株）


-かんなJ_:--%%-Mule: *Mew message*0    Mon Sep  8 6:
Please answer y or n.  <LI> 1997/10/31 <a href="dpsm
```

図 5.4 承認の様子

5.8　差替支援部

要件と実装

承認支援部から送られた差分情報を受け取り、この情報を元に、文書管理部に対して編集行為を反映させるための操作を指示する。満たすべき要件や実装は、差分登録支援部と

同じである。

5.9 文書管理部

文書管理部は、差分連絡支援部や差替支援部からの命令に従い、受け取った文書をデータベースへ保存する部分であり、本システムの中核部分である。以下では、まず文書管理部の構成について述べ、その後では、文書管理部の挙動について、外部からの参照要求、およびデータベースへの保存に注目して述べる。

5.9.1 構成

文書管理部は図 5.5 のような構成による。

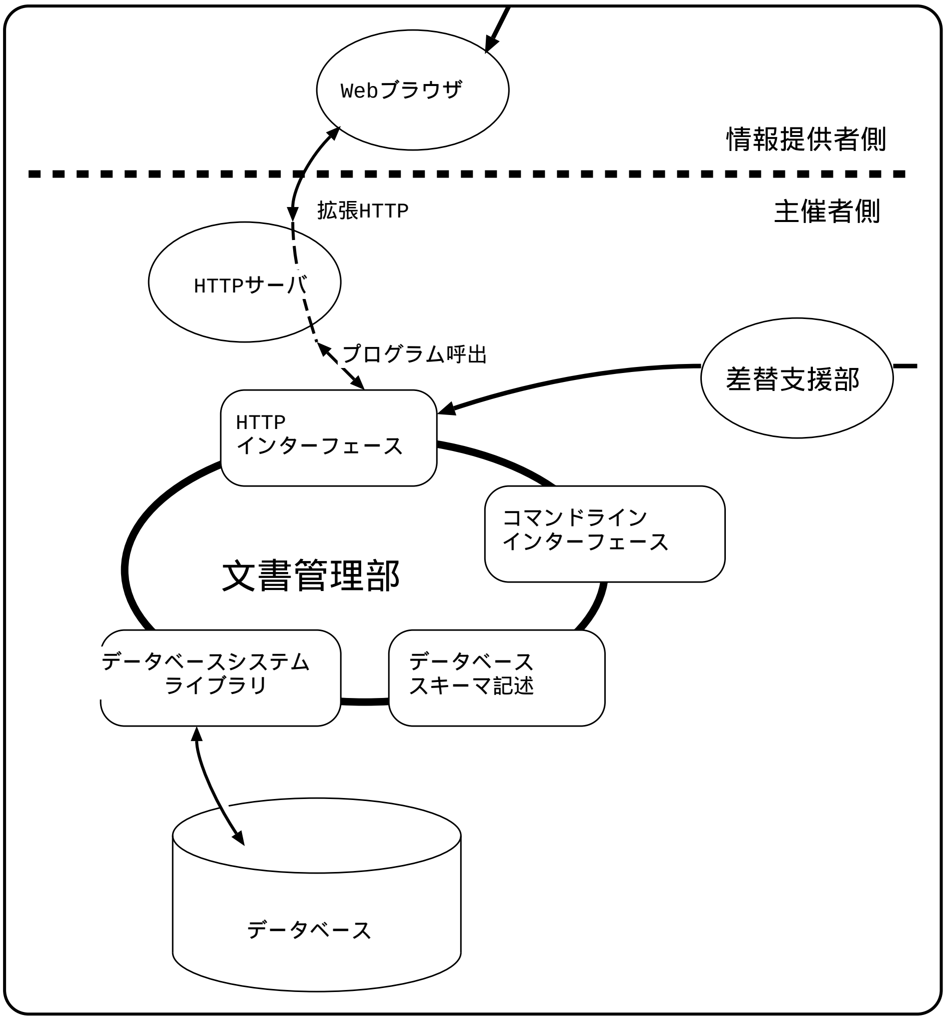


図 5.5 文書管理部の構成

- データベースシステムライブラリ
- データベーススキーマ定義
- HTTP インターフェース
- コマンドラインインターフェース

データベースシステムライブラリ データベースに関する操作を提供するライブラリであり、Java ベースのオブジェクト指向データベースシステムである Object Design 社の ObjectStore PSE Release 1.0.3 [4] を用いて開発を行なった。

データベーススキーマ定義 データベース内に格納する永続オブジェクトを示すものであり、今回はテキストに関するハイパーテキスト構造を表すクラスを用意した。将来、この部分にさまざまなデータタイプに対するスキーマ定義を用意し、また、これらの基本的な定義を組み合わせることにより、表組のような構造を表すことが可能になる予定である。

HTTP インターフェース HTTP インターフェースは差替支援部を実装した WWW ブラウザと通信する部分であり、独自の拡張を加えた HTTP プロトコルを理解する。今回の実装には、servlet(JSDK1.0.1[Sys]) を用いている。servlet は HTTP サーバから呼び出され、HTTP サーバを仲介して差替支援部と通信を行なっている。

コマンドラインインターフェース 主催者が用いる管理用のコマンドを受け付ける。初版となる文書の登録、承認/非承認による文書のバージョンアップや消去のコマンドを用意した。

5.9.2 外部からの参照要求

情報提供者およびその他の参加者からのデータの参照要求に対しての、文書管理部が行なう振るまいについて述べる。協同文書は WWW を通じて参照されるため、参照には HTTP プロトコルが使用される。http サーバに対して協同文書を表す URI が指定されると、http サーバは文書管理部を servlet プログラムとして起動する。servlet は、クライアントとの通信に必要な HTTP のメソッドを自由に拡張することができる (カスタムメソッドと呼ばれる) ので、拡張 HTTP の実装に都合が良い。協同文書を表す URI から servlet プログラムが呼び出される様子を図 5.6 に示す。

次に、文書管理部は、指定された URI をキーとしてデータベースに対して検索を行なう。「窓つき文書」はデータベース中では各構成要素に分割されて保存されているが、これらをデータベース中から取り出して再構成することで、指定された「窓つき文書」がデータベースから取り出される。

「窓つき文書」は単独では意味の取れない文書となるので、参照されるときには窓を通して向こう側が見えるようにする必要がある。これは、窓と窓の対応づけをたどることで

---

[4] http://www.odi.com/

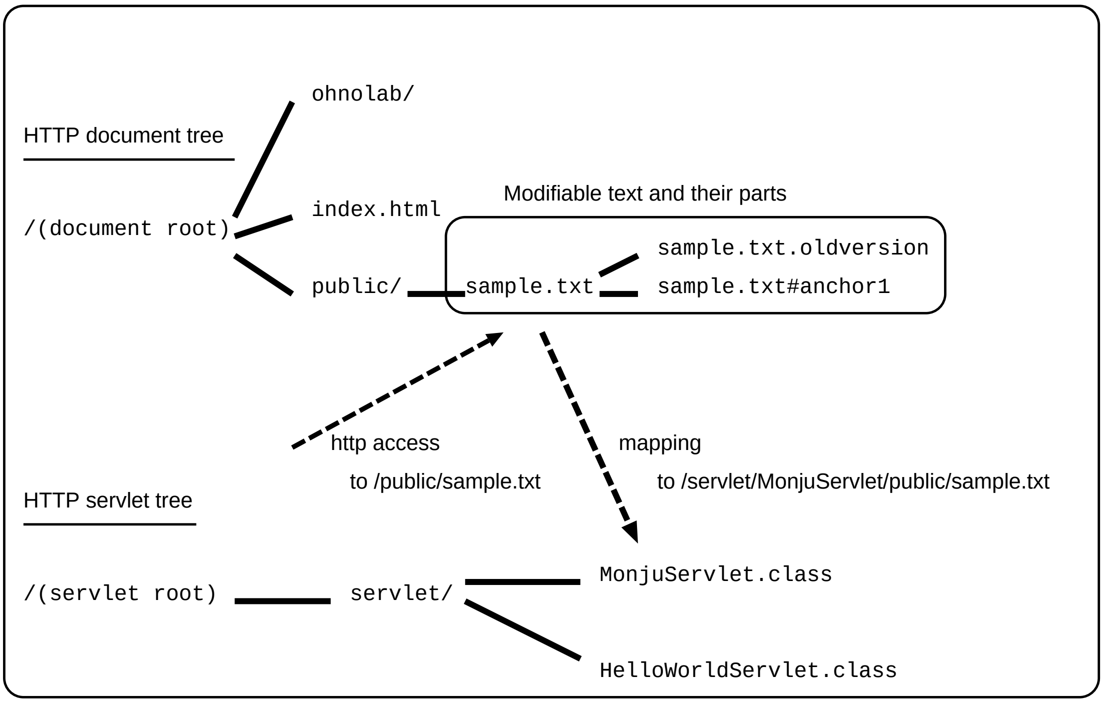


図 5.6 協同文書へのアクセスによる servlet の呼び出し

参照元の文書に到達し、窓に対応する参照部分を取り出すことで実現できる。窓の向こう側にもまた窓がある場合には、以上を再帰的に繰り返すことによって、「窓つき文書」が表す協同文書を完全に取り出すことができる。同時に、著者情報などの属性値の取得を行ない、協同文書と共に送出する。

5.9.3 データベースへの保存

文書管理部は、差分登録支援部や差替支援部からの命令を受けると、データベーススキーマにしたがって命令に対応するオブジェクトの生成や破棄を行ない、結果をデータベース中に保存する。保存の際、文書の各構成要素には、著者が誰かを示す情報、キャッシュのために必要となる文書の作成時刻を示す情報、などの属性を与える。

文書管理部が理解する拡張 HTTP プロトコルのメソッドのうち、差分登録支援部や差替支援部が発する命令は、以下の 3 種類である。

- POST メソッド
- ANCHOR メソッド
- COMPOSE メソッド

POSTメソッド

POSTメソッドは、同時に送るデータを、与えられたURIで指示されるリソースの付随情報として受け入れることをサーバに要求するメソッドである。すなわち、サーバは与えられたURIに関連する新たなリソースを作成し、届いたデータを格納する。

情報提供者が協同文書に対して、新たに書き加えた部分をデータベースに格納するために使用する。

文書管理部は、新たに書き加えられた部分を内容とするオブジェクトを新たに作成し、そのオブジェクトに適当な名前を新たに割り当てる。この名前は、新たに書き加えられた部分の内容を外部から参照するための識別子であるURIに相当する。

ANCHORメソッド

指定されたリソースに対してアンカーを付加するためのメソッドである。リソースを示すURIの他に、アンカーの範囲を示す情報を必要とする。

情報提供者が協同文書中の既存の部分を再利用するために用いる。

文書管理部は与えられたURIに対応するオブジェクトに対して指示された範囲を参照するためのオブジェクトを新たに生成する。対象となるオブジェクトがすでに多数の分割された部分から構成されている場合は、構成要素である分割部分を先頭から順に調べることによって、指示された範囲を取り出せるようなオブジェクトを生成する。

新たに生成されたオブジェクトに対して、適当な名前を新たに割り当てる。この名前は、付加されたアンカーの内容を外部から参照するための識別子であるURIに相当する。

COMPOSEメソッド

既存のリソースを直線的に並べた、新たなリソースを生成して参照可能にすることを要求するメソッドである。構成要素となるリソース群のURIの列を必要とする。

情報提供者が、既存の文書や、POSTメソッドやANCHORメソッドを用いて新たに生成または部分指定した文書を組み合わせて、新たな文書を生成するために用いる。

文書管理部は、指示された順にリソース群をまとめて参照するオブジェクトを新たに生成し、そのオブジェクトに適当な名前を新たに割り当てる。

5.9.4 日本語への対応

バイト列をそのまま文字とみなすことができるアスキー文字セットに対して、日本語は一般的に多バイト文字である。このため、テキストの範囲を表すアンカーの付加に関して、注意を払う必要がある。

テキストに対してアンカーを付加するときは「n文字目からm文字目まで」のように指定される。このとき、対象となるテキストがアスキー文字セットのみから成る場合は、「nバイト目からmバイト目」のように解釈しても構わない。一方、対象となるテキストに

日本語が含まれる場合、それは成り立たない。さらに、符号化の方法によっては日本語の文書には符号化の境界を表すエスケープシークエンスが含まれるので、日本語の文書に対して単純にバイト数に依存したアンカーの付加を行なうと、参照の際に思わぬ結果を招く。従って、日本語を含むテキストに対してアンカーを付加するときは、日本語の1文字の単位に配慮する必要がある。この配慮は、文書管理部および差分作成部で必要となる。

今回の文書管理部の実装で用いたJava言語では、内部コードとして日本語の1文字とアスキー文字の1文字が同等に扱われるため、ネットワークとの入出力のみに注意すれば良く、日本語への対応は比較的簡易である。

差分作成部の実装例ではemacsの多国語化対応版であるmule用に拡張されたemacs-lispを利用したが、バッファ内の編集の際に、日本語の1文字とアスキー文字の1文字を同等に扱うための工夫をほどこして実装を行なった。

5.9.5 拡張HTTP

文書管理部は、差分登録支援部とのインターフェースとして、拡張されたHTTPを用いる。これは、WEBDAV-WGによって提案されたリクエスト[GJWF+97]を参考にし、HTTP/1.1[FGM+97]にプロトコルを追加する形で拡張を行なった。

文書管理部が差分登録支援部からの命令に対してどのように振舞うかはすでに述べたが、ここでは新たに追加されたメソッドのみを取り上げて、その定義を以下に簡単に示しておく。

ANCHOR 指定されたリソースに対してアンカーを付加するためのメソッドである。

リソースを示すURIの他に、アンカーの範囲を示す情報を必要とする。アンカーの付加に成功した場合は、付加したアンカーの名前を返す。

COMPOSE コンポージットとは、分割された文書を参照するために、文書の構成要素となるリソースを直線的に並べたものである。指定されたリソースからコンポージットの参照を可能にするためのメソッドである。

リソースを示すURIの他に、構成要素となるリソース群のURIの列を必要とする。

実行の成否を返す。

なお、これらのメソッドについては、既存のメソッドの解釈を変えることによって既存のメソッドに代用できる可能性があるため、議論の余地がある。

第6章

評価

本章では、本システムの評価を行なう。本システムを用いた運用経験から得られた運用結果に基づく評価、および要件として挙げた機能の達成を設計の点からの評価を行なう。また、本システムの応用例を示す。

6.1　システム運用と評価

本節では、本システムを用いた運用例を2件示す。行なった2件の運用は以下のとおりである。

- 論文募集情報の集約
- 論文の校正

それぞれの運用の概要、および運用結果の評価について報告する。

6.1.1　論文募集情報の集約

概要

筆者らは本システムのプロトタイプを用いて研究室内で方々から寄せられる論文募集や講演参加者募集の情報を収集する機構を運用している [門間 97a]。

この運用システムでは、情報は追加されるのみであることを前提としている。また、作業モデル中、差分登録の過程は禁止されており、実際の情報の登録は差替過程で行われる。結果、仕組みとしては単純になり、実装の一部が用いられるに過ぎないが、作業モデルやデータベース登録方法の評価は可能と考えている。

情報提供者は自分が持っている論文募集や講演参加者募集の情報を主催者に電子メールで送付する。以下、作業モデルにしたがって作業が行なわれ、最終的に送付された情報はデータベースに格納され、WWW を通じて閲覧することができる。

類似の試みとしては、ネットニュースの特定のニュースグループに集められた会告記事の要約を自動作成するシステム [佐藤 95] が挙げられる。

表 6.1 論文募集情報の集約における運用結果

期	第一期	第二期
期間	1996年11月~1997年3月	1997年4月~現在
計算機支援	なし	承認支援部・差替支援部
承認要求の数(うち、却下の数)	93(7)	174(4)
承認・差替過程に要した時間(日)	7.4	1.7

運用は1996年11月より開始され、1997年3月までは特に計算機支援を行なわず、主催者は作業を手作業で行なった。この期間を第一期とする。その後、1997年4月より、承認支援部および差替支援部を用いて主催者に対する計算機支援を行なった。この期間を第二期とする。

運用結果と評価

現在までの運用の結果、267件の承認要求が寄せられている。内訳は、第一期が93件、第二期が174件である。このうち、11件が内容の重複などの理由で却下され。残りの256件が登録された。却下の内訳は、第一期が7件、第二期が4件である。

システムによる主催者の負担軽減の評価を行なうために、承認過程および差替過程に要した時間を計測した。個々の承認要求において、主催者が承認要求の電子メールを受け取った時刻および差替過程においてデータベースへ登録が行なわれた時刻を記録し、その時刻の差から求められた時間を承認過程および差替過程に要した時間とみなした。この時間を却下されずに承認された全ての承認要求に対して求め、さらに平均を求めると、第一期が7.4日、第二期が1.7日であった。以上の結果を表6.1にまとめる。

第一期では、主催者は作業を手作業で行なっていたが、大量に承認の要求が寄せられて作業が滞ることがあった。しかし、第二期では、承認・差替作業に要する時間は大幅に短縮され、作業の滞りは解消されている。したがって、主催者が更新作業のために必要とする労力の軽減に効果があったといえるであろう。

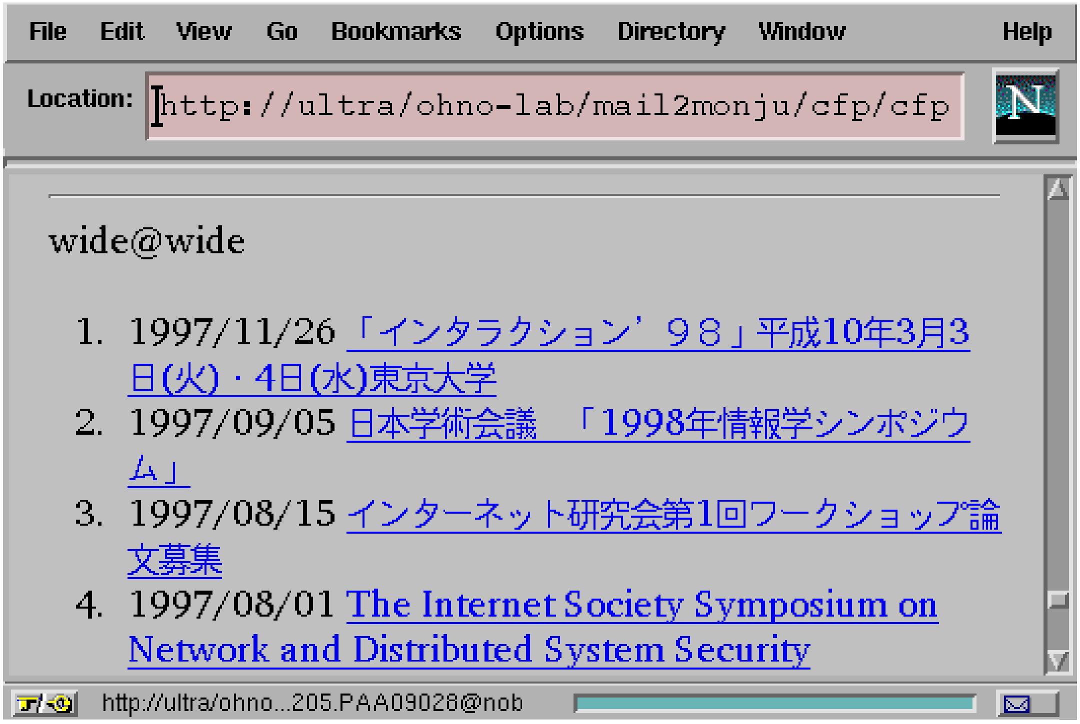
収集された情報を図6.1に示しておく。

文書の管理方法および文書管理部の実装に対する評価は、協同文書への追加だけでなく、さまざまな執筆形態で行なう必要がある。

6.1.2 論文の校正

概要

校正、いわゆる「赤入れ」は、インターネットメッセージで見られるようなおおまかな文書の引用に比べて文書の編集の単位が細かいため、複雑な編集行為の典型例と言える。



http://ultra/ohno-lab/mail2monju/cfp/cfp

wide@wide

1. 1997/11/26 「インタラクション’９８」平成10年3月3日(火)・4日(水)東京大学
2. 1997/09/05 日本学術会議　「1998年情報学シンポジウム」
3. 1997/08/15 インターネット研究会第1回ワークショップ論文募集
4. 1997/08/01 The Internet Society Symposium on Network and Distributed System Security

http://ultra/ohno...205.PAA09028@nob

図 6.1 収集された情報

結果として、文書内には複数の著者による執筆部分が複雑に入り混じる。このような場合に文書管理部がどのような挙動を示すかを調べ、評価を行なう。

本論文の第 1 章を、本システムを研究室内で運営することにより、校正を行なった。

実験環境

主催者側、情報提供者側それぞれの実験環境を以下に示す。

主催者側の環境　WWW サーバには、Apache Digital 社の apache-1.1.3[1] に、apache 用の servlet モジュール mod_servlet.c を組み込んで使用した。文書管理部には、JSDK 1.0.1 および JDK 1.1.5 を用いた。マシンは Sun UltraSparc、OS は Solaris 2.5.1 である。

主催者が用いる差替支援部および承認支援部に関しては mule-2.3(Emacs 19.28.1) 上で動作する差替支援部およびメールリーダである mew-1.91[2] に組み込まれた承認支援部を用いた。

情報提供者側の環境　情報提供者が用いたマシンと OS、差分作成支援部で用いたエディタ、差分登録支援部として用いた WWW ブラウザは、表 6.2 のとおりである。

---

[1] http://www.apache.com/
[2] http://www.mew.org/

表 6.2 情報提供者側の環境

マシン	OS	差分作成支援部	差分登録支援部
PC-AT(Pentium)	BSD/OS 2.1	Mule 2.3(Emacs 19.34.3)	w3-2.2.25
PC-AT(Pentium)	BSD/OS 3.0	Mule 2.3(Emacs 19.34.3)	w3-2.2.25
SONY NEWS5000	NewsOS 4.2.1R	Mule 2.3(Emacs 19.28.1)	w3-2.1b
Sun UltraSparc	Solaris 2.5.1	Mule 2.3(Emacs 19.28.1)	w3-2.2.25

運用結果と評価

6 人の情報提供者によって、12 件の差分登録が行われた。うち主催者によって 10 件が承認され、1 件が却下された。これは、同一の情報提供者がほぼ同一内容の変更を 2 度行ったため、一方を却下したものである。残りの 1 件は、情報提供者が登録を行ったものの、連絡用の電子メールの送付に失敗したために主催者に承認要求が届かず、承認が行われなかったものである。

運用の結果、できあがった文書が第 1 章である[3]。承認が 10 件であったから、10 回の変更が行われたことになる。各回の変更の詳細を図 6.3 に示す。「情報提供者」は、書換えを行った情報提供者が誰であるかを示す。情報提供者のうち、A は原版を用意した主催者であり、B～F の 6 人が情報提供者である。「編集行為の数」は、変更の際に行われた「挿入」「削除」などの編集行為の数であり、送付された差分から計算した。この数値が大きいほど、大幅な編集を加えたことを示す。「協同文書の文字数」は、その版の協同文書に含まれる文字数であるが、この数値には大きな変化はない。協同文書は多数の既存の文書の部分 (「窓つき文書」の「窓」) を組み合わせで構成されるが、「分割数」は、その版の協同文書を構成する部分の数である。版を重ねるに従って、協同文書は多数の構成要素に分割されることが示されている。最終版では、約 1,000 文字の文書が約 250 の部分に分かれており、協同文書内で各人が執筆した部分がいかに複雑に入り組んでいるかがわかる。

協同文書の各版において、各情報提供者の執筆部分が占める文字数の推移を図 6.2 に示す。縦軸が版、横軸が文字数、枠内に書かれたアルファベットは情報提供者を表す。最上段が主催者が用意した原版であり、以下、版を重ねるに従って、情報提供者の占める割合が高くなり、最終版では原版の 50%程度が書き換わっていることがわかる。

各情報提供者が占める文字数の推移に着目すると、情報提供者 C が執筆した部分は多くが最終版まで残っており、有用な書換えが多かったことがわかる。一方、情報提供者 D が執筆した部分の多くはすぐに他の情報提供者によって書換えられており、あまり重要な変更とみなされなかったことがうかがえる。このように、協同文書内で各人が文書の作成にどの程度貢献したかを示す一つの指標とすることができると考えられる。

寄せられた意見として、編集の単位が細かすぎる点が挙げられる。英単語のスペルミスの訂正から、句読点、文書中の空白の数、改行位置にまで、情報提供者が新たな著者として登録されるのは管理すべきの程度を超えているとの指摘である。実際、10 個の版の合計

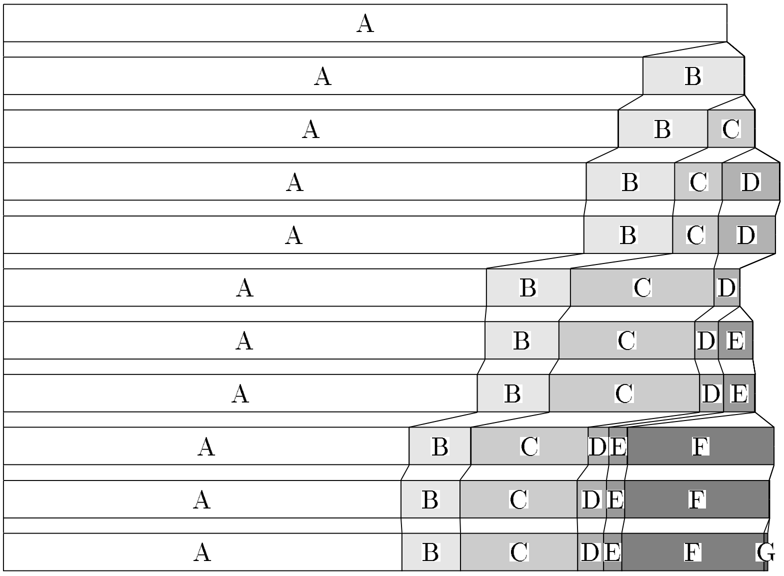
---

[3] 本論文は TeX で整形されている。得られた協同文書の TeX を用いた整形は筆者が行なった。

表 6.3 各変更の詳細

版	情報提供者	編集行為の数	協同文書の文字数	分割数
原版	A	1	957	1
1 回目	B	60	980	60
2 回目	C	20	994	69
3 回目	D	42	1027	107
4 回目	D	4	1021	107
5 回目	C	54	974	134
6 回目	E	37	991	154
7 回目	C	17	994	153
8 回目	F	174	1019	252
9 回目	D	11	1013	254
10 回目	G	4	1011	255

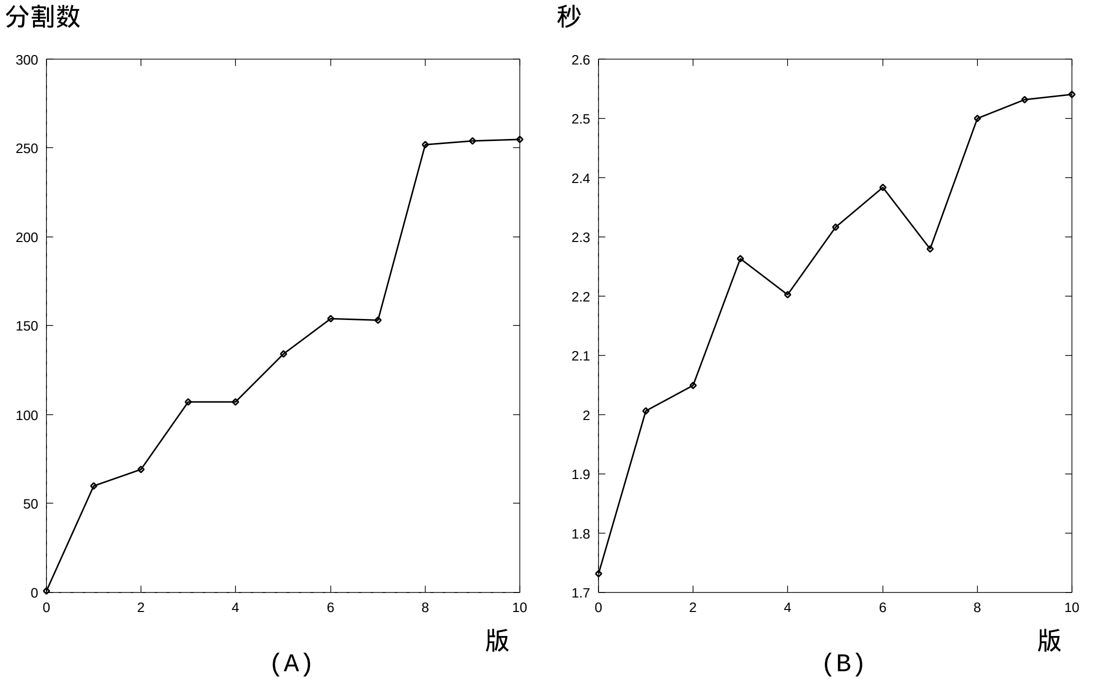
図 6.2 各情報提供者の執筆部分が占める文字数の推移



約500の編集行為のうち、改行位置の変更のみの編集が約50を占めている。

実行時間について　協同文書中の分割数が多くなった場合の、文書管理部が行なうデータベースからの協同文書の所得にかかる時間について計測した結果を示す。

各バージョンにおける分割数の推移は表6.3で表したが、これをグラフにしたものが図6.3(A)である。一方、参照に対して、各バージョンをデータベースから取り出すために要した時間を図6.3(B)に示す。二つのグラフを比較すると、分割数が増大するにしたがって取り出しに要する時間も増大していることがわかる。



(A)　(B)

図6.3 各バージョンの分割数と、データベースからの取得にかかる時間

グラフ中の数値では、取り出しにかかる所要時間は最大でも2秒程度であるため、取り出された文書のネットワーク転送にかかる時間に比べて大きな数値ではない。しかし、さらに分割数が多くなった場合は無視できない時間がかかることが予想される。したがって、分割数の増大を抑制する仕組み、もしくは次章で述べるようにキャッシュを用いて文書の取得にかかる時間を短縮する仕組みが必要となるであろう。

なお、論文の校正については、アノテーション機能を採り入れることによって、より効果的なシステムになることが期待できる。

6.2　評価

本研究の主目的は、情報の集約作業における主催者の負担を軽減することである。この点については、前述の運用実験によって効果の感触を掴んでいるが、より正確な評価に関

しては長期にわたる本格的な運用実験が必要である。

6.2.1　協同執筆支援システムとしての評価

レビュー機能

前述した実験では、論文の校正を題材として、各情報提供者の執筆部分が複雑に入り混じる場合に文書管理部がどのような挙動を示すかを調べた。

しかしながら、論文の校正のような作業は、第2章で述べたように、一人の執筆者が共同研究者や指導者の意見や添削の下に文書の作成を進める形態、すなわち、レビューの繰り返しによる協同執筆形態の典型例であり、文書全体や一部分に対するコメントづけの機能が重要になる。本システムでは、執筆作業を分担する形態を主にサポートするため、本システムを単独で用いた場合には、論文の校正行為の支援は不十分であると言える。

本システムにおいて、コメントの付加のようなレビューを行なう機能を導入する方法について、以下に述べる。

既存のアノテーションシステムとの連携

既存のアノテーションシステムとの連携によって、コメントの付加をサポートする方法について述べる。

本システムでは、新たに開発したハイパーテキストシステムがWWWとのインターフェースを備えており、本システムを利用しない一般のユーザからは協同文書は一般のWebページと同じに見える。したがって、WWWのページを対象とした既存のアノテーションシステムを、改変することなくそのまま利用できる。

アノテーションシステムの中には、文書に対して付加されたコメントを文書の末尾などに直接書き込むことによってWebページへのコメントの付加機能を実現しているシステムもあるが、その程度の機能であれば本システムでも実現可能であり、連携に値しない。

ComMentor[RWP95]のように、アノテーションを文書とは別のアノテーションサーバに保存する仕組みのシステムであれば、文書の全体や一部分に対するコメントを元の文書を改変することなく付加することができる。論文の校正のような作業形態の場合、主催者はこのようなアノテーションシステムと本システムとを併用することにより、十分なサポートを受けることができると思われる。

このような仕組みのアノテーションシステムでは、コメントの元の文書が改変されたとき、付加されていたコメントが意味をなさなくなるという問題点がある。しかし、本システムでは、各バージョンにはそれぞれ固有のURIが与えられるため、主催者が承認によるバージョンの更新を行なっても、バージョン固有のURIに付加されたコメントは維持されている。

ただし、ComMentorの利用には専用のブラウザを必要とするため、情報提供者は必要に応じて専用ブラウザと本システムの差分作成部とを使い分ける煩わしさが生じる点が欠点である。

本システムにコメント付加機能を導入する方法

本システムを拡張し、協同文書に対してコメントを付加する機能を導入する方法について述べる。

文書および文書に対するコメントを管理することは、ハイパーテキストシステムの基本的機能である。本システムの各部分のうち、文書管理部ではデータベースにコメントを格納しておくことは可能である。また、文書管理部のプロトコルであるHTTPもコメント付加に対応している。HTTPで定められている通信方式であるPOSTメソッドは、本来、Webページに対するコメント付加のための利用を想定したものである。その解釈はサーバおよび対象となるWebページに委ねられているので、文書管理部においてPOSTメソッドによって送信されたコメントを解釈する処理を取り決め、実装すれば良い。

しかし、差分作成部においては、2つの文書をplaintextと見立てて差分を取る機能しか用意されていないので、コメントを扱うことができない。したがって、情報提供者がコメントを付加するためには差分作成部の拡張を行なう必要がある。具体的には、差分作成部に従来の文書レビューシステムを参考に、元の文書とは別のバッファでのコメントの書き込みなどの機能を備えれば良い。

差分作成部は、情報提供者が使用しているエディタとの連携によって支援を行なうが、このような拡張を行なうにはエディタがやや高機能でなくてはならない点が問題である。

アノテーションシステムやConferencing Systemとの比較

第3章で述べたように、WWW上での協同執筆作業の支援のために、アノテーションシステムやWeb Conferencing Systemと総称されるシステムが数多く存在するが、これらは協同執筆の形態のうち、文書のレビューに着目したものである。対して本システムは、執筆作業を分担するものである。したがって、目的は異なるが、これらのシステムの組み合わせにより、幅広い形態の協同執筆作業を支援することができる。

すなわち、前述のように、本システムとアノテーションシステムの連携により、より一般的な協同執筆作業形態をサポートすることができる。また、Web Conferencing Systemが提供するコミュニケーション環境は、協同執筆作業の支援において、共著者間での執筆プランのためのコミュニケーションにおいて重要な役割を果たす。

WEBDAV-WGの試みとの比較

WEBDAV-WGの試み[GJWF+97]は、WWWにおける協同作業の構築を主目的とはしていないものの、複数人でのWebページの編集を考慮に入れており、協同執筆作業を支援するシステムと言える。問題解決の手段としてHTTPのメソッドの拡張を行なっている点も本システムと同様である。

しかし、WEBDAVでは、HTTPをあくまでHTMLで書かれた文書を転送するプロトコルとしてとらえている。そのため、ハイパーテキストのアンカーやリンクなどは、全て情報提供者がHTMLを用いて記述するものであるという立場を取っている。したがって、

通信の仕組みは単純になる。

情報提供者が WWW サーバに対して執筆内容を送付する場面を例にとると、本システムを用いた場合は、編集内容の複雑さに応じてブラウザと WWW サーバの間で多数のやりとりが行なわれるのに対し、WEBDAV の枠組みを用いて協同文書の編集を行なう場合は PUT プロトコルによる文書の送付が 1 度行なわれるのみである。したがって、通信に不安定な回線を用いる場合は、WEBDAV の枠組みを用いた場合のほうが通信形態として優れていると言える。本システムでは、この点を補うために、情報提供者が差分登録過程に失敗した場合でも、連絡過程を経由して差分の登録を行なうことができる仕組みを用意している。

また、WEBDAV では、文書に対する部分ロックのための操作を規定するなど、幅広い形態の協同作業へのサポートを目指している。この点では、本システムにおいても、支障のない範囲で必要に応じて WEBDAV に追随する予定である。

しかし、WEBDAV では HTML をベースとしているため、ハイパーテキスト機能としては次小節で述べるように本システムに比べて劣っている。そのため、ハイパーテキスト機能に基づくバージョン管理の機構も明確になっていない。

6.2.2 情報サービスシステムの機能評価

ここでは、目的のための手段として用い、開発を行った、新たな協同文書の保存や協同作業のための情報サービスシステムについて、第 3 章で掲げた要件に関して、関連研究と比較しながら評価を与える。

文書の一部分を取り出して参照できること　WWW 上のリソースに関して、元の情報の内容を直接変更せずに、第三者が部分参照のためのアンカーやリンクを付加する試みに関しては、DHM/WWW[GBS97] に詳しい。これは、DHM サーバと呼ばれる、アンカーやリンクを管理するサーバと Java アプレットを用いて通信を行い、アプレット内で Web ページとリンク情報とを結合させるものである。

本システムでは、文書管理部の設計に関して DHM と同じモデルに基づいており、DHM サーバと文書管理部の機能の差は小さい。

文書の部分同士を組み合わせて、新たな文書とすることができること　これはハイパーテキストの分野で Path あるいは Tour と呼ばれる仕組みである。Path は個人が異なるハイパーテキストをまとめて取り扱うために考案された。WWW における Bookmark は個人的に多くの Web ページをまとめて扱うことができ、Path の一種といえる。また、Path を作成するための Authoring Tool 、Path を閲覧するためのブラウザの実装を含めて、Path の実装と称する。

WWW において Path を実装した例は Deckscape、Walden's Path[FSM^{+}97] などがある。これらのシステムは、個人での情報整理や、教育機関での教材として用いることを目的としているため、協同作業環境の実現を目的とする本研究とは目的が異なるが、Path としての機能面では本システムは強力である。特長を以下に述べる。

- Path の構成要素として、ページ全体のみでなく、ページの部分をも指定できる。
- 構成した Path を個人的空間内で利用するにとどまらず、公開することができる。
- 他者が構成した Path の全部あるいは一部分を利用して、新たな Path を構成することができる。

文書の部分ごとの著者を取得できること　この機能によって、協同文書の著者が複雑に入り混じった場合でも、文書中における各人の執筆量を求めることができ、それによって各人の貢献度のような値を定めることができると思われる。

この機能の性能は、差分作成部でいかに良い差分を生成できるかに大きく依存する。運用実験では、差分作成部は文字単位の細かい粒度で著者を区別できる差分を生成することが確認されている。反面、粒度が細かすぎるために、文書中の空白の数のような、意味の薄い変更にも忠実に著者情報が示されてしまうことも指摘されているのは前述のとおりである。

なお、WWW クライアントから Web ページの部分ではなく、全体の属性値を取得する方式に関しては [GJWF$^+$97] がある。

6.3　応用例

本システムの実装の充実によって実現可能となる応用システムについて、具体例を挙げて説明する。

6.3.1　研究室内物品管理システム

著者らの研究室では、構成員間の情報共有システムの一つとして物品購入・管理システムの運用を行っている [野田 97]。このシステムでは、物品の情報が記載されている台帳への登録が手作業によって行われていたために煩わしさが感じられると共に、誤登録や事故がたびたび発生している。この問題点の解決方法として、本システムを導入する方法を示している [門間 97b]。

台帳は、エントリごとに物品名、購入店名、価格などがならんだ表の形式になっている。表組のような構造を持った文書に対するスキーマを定義し、表中の各欄に対しての編集を可能にすることによって、多人数が台帳を細かく管理することができると期待される。

第7章

課題

今回は、研究室内での運用によって本システムの有用性の評価を行なった。しかし、広域ネットワークにおいて、広く利用するための情報システムとしては、不十分である点も多い。以下に課題を列挙する。

文書更新時の著者情報の取得

本システムでは、著者の協同文書における権利の範囲を明確にできる特徴を持っている。しかし、実際には著者が誰であるかを厳密に特定することは容易ではない。

まず、更新された文書の登録はWWWを通じて行なわれるが、接続の相手を特定できない場合が多い。WWWでの通信に用いられるHTTPプロトコルにはFrom:ヘッダが規定されており、このヘッダに電子メールアドレスを格納することができる。しかし、From:ヘッダの付加は必須ではないため、通常用いられることは希である。現在の実装では、差分登録支援部の実装であるw3.elの中で、From:ヘッダの利用を義務づけており、文書管理部ではこの情報を著者情報として格納している。

一方、文書の変更には電子メールを通じた承認要請を必ず伴う。電子メールの場合は、送出者を示すFrom:ヘッダは必須であるため、WWWに比べて相手を特定しやすい。したがって、承認要請の電子メールアドレスから取得した情報を文書の著者情報とすることも可能である。しかし、この場合も正しいアドレスが記述されている保証はないため、厳密な認証を求めるのであれば、何らかの新しい仕組みを導入することが必要となる。

協同文書を取り出す際のキャッシュ

協同文書は、データベース中に「窓つき文書」の形で、多数の部分に分割されて保存されている。従って、協同文書を取得するためには、データベース中から多数の構成要素をハイパーテキスト構造に従って再構築する必要がある。このため、WWWにおける通常のファイルで表された情報を取得するためにかかる時間に比べると、協同文書を取得するためには長い時間がかかる。通常、協同文書はWWWを用いて取得されるので、協同文書の構成要素が全て単一の計算機内に存在するのであれば、文書の分割数が大きくならない

限り、ネットワーク転送にかかる時間に比べてそれほど大きな影響は与えないことは前章で示した。しかし、協同文書の構成要素として外部の計算機中の文書が含まれる場合には、複数の計算機にアクセスする必要があるため、協同文書の取得にかかる時間は大幅に長くなる。

これがユーザにとって無視できる程度の時間か、あるいはストレスとなる時間かは運用実験を行なっていないため、現在は未知であるが、無視できる程度でない場合、協同文書の取得にかかる時間を短くするためのキャッシュの仕組みが必要になるであろう。

差分作成支援部の動作環境の拡大

今回は、ほとんどの部分の実装に対するプログラミング言語としてEmacs LispやJavaを用いている。これらの言語は特定のプラットフォームに依存しないので、多くの環境で動作することが期待できる。

しかし、今回行なった差分作成支援部の実装は、筆者が日常的に利用しているエディタであるemacsの機能に大きく依存しており、他のエディタで同等の機能を提供できる保証はない。情報提供者は、協同文書の編集を普段使い慣れているエディタで行なうことを望むことが予想されるので、差分作成支援部の動作環境を広げる必要がある。

ユーザ検証機構の充実

今回の文書管理部では、差分作成支援部において生成された差分のうち、著者が独自に書いた部分に対して電子署名をほどこすことにより、著者情報の管理に利用するだけでなく、主催者による文書の改ざんを防止する機構の導入を予定している。

しかし、改ざんが行なわれた場合は、確かに検証を行なえば改ざんの事実を明らかにすることはできるのだが、文書がアクセスされるたびに検証を行なうことは、処理速度の点、および署名用の鍵配布サービスなどの整備状況の未熟さの点から現実的ではない。

したがって、本システムにおける改ざん防止のための電子署名は、現在の環境ではその威力を十分に発揮することができない。電子署名をとりまく環境の一層の整備が求められる。

第8章

おわりに

コンピュータネットワークの上の人々が持つ情報を、協同執筆によって集積するための作業の流れを提案し、途中の過程の各作業を支援することで協同作業者の労力を軽減する仕組みを示した。また、協同文書を保存するためのデータベースの構築方法と、データベースを操作する機構を示し、設計および実装を行なった。さらに、システムを研究室内で運用することによって、作業の際の主催者の労力が軽減されることを示した。

課題としては、研究室内でなく、広域ネットワークにおいて運用を行うために必要な機能の付加および実装が挙げられる。

謝辞

研究を進めるにあたっての丁寧な御指導および精神的なケアをしていただいた大野浩之講師に心から感謝致します。

また、崖っ淵の状態からはい上がるための研究の進め方について多数の助言を与えて下さった国立がんセンターの中川晋一氏、実験に協力してくださった大野研究室の皆さんに、この場を借りてお礼を申し上げます。

参考文献

[BLC95] T. Berners-Lee and D. Connolly. Hypertext Markup Language - 2.0, RFC1866, November 1995.

[BLCGP92] Tim Berners-Lee, Robert Cailliau, Jean-Francis Groff, and Bernd Pollermann. World-Wide Web: The Information Universe. *Electrinic Networking: Research, Applications and Policy*, Vol. 2, No. 1, pp. 52–58, 1992.

[BLFF95] T. Berners-Lee, R. Fielding, and H. Frystyk. Hypertext Transfer Protocol – HTTP/1.0, RFC1945, 1995.

[CRHH95] Lesile Carr, David De Roure, Wendy Hall, and Gary Hill. The Distrivuted Link Service: A Tool for Publishers, Authors and Readers. In *Proceedings of Fourth International World Wide Web Conference*, Boston, Massachusetts, USA, December 1995. The World Wide Web Consortium.

[D+95] John December, et al. *The World Wide Web Unleashed.* Sams.Net Publishing, 2nd edition, April 1995. http://www.december.com/works/wwwu.html.

[DH] Jim Davis and Dan Huttenlocher. CoNote (Annotation) homepage. http://dri.cornell.edu/pub/davis/annotation.html.

[FGM+97] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, and T. Berners-Lee. Hypertext Transfer Protocol - HTTP/1.1, RFC2068, January 1997.

[FSM+97] Richard Furuta, Frank M. Shipman, Catherine C. Marshall, Donald Brenner, and Hao wei Hsieh. Hypertext Paths and the World-Wide Web: Experiences with Walden's Paths. In *Hypertext '97 – Eighth ACM Conference on Hypertext*, pp. 167–176, Southanmpton UK, April 1997. ACM, ACM Press.

[GBS97] Kaj Grønbæk, Niels Olof Bouvin, and Lennert Sloth. Designing Dexter-based hypermedia services for the World Wide Web. In *Hypertext '97 – Eighth ACM Conference on Hypertext*, pp. 146–156, Southanmpton UK, April 1997. ACM, ACM Press.

[GHMS94] Kaj Grønbæk, Jens A. Hem, Ole L. Madsen, and Lennert Sloth. Cooperative Hypermedia Systems: A Dexter-Based Arhitecture. *CACM*, Vol. 37, No. 2, pp. 64–74, February 1994.

[GJWF+97] Y.Y. Goland, Jr J. Whitehead, A. Faizi, S.R. Carter, and D. Jensen. Extensions for Distributed Authoring on the World Wide Web – WEBDAV, Nobember 1997. INTERNET DRAFT(Expires April, 1998).

[GT94] Kaj Grønbæk and Randall H. Trigg. Design issue for a Dexter-based hypermedia system. *CACM*, Vol. 37, No. 2, pp. 40–49, February 1994.

[HS94] F. Halasz and M. Schwartz. The Dexter hypertext reference model. *CACM*, Vol. 37, No. 2, pp. 30–39, February 1994.

[LB95] D. LaLiberte and A. Braverman. A protocol for Scalable Group and Public Annotations. *Computer Networks ISDN Systems*, Vol. 26, No. 6, pp. 911–918, June 1995. http://www.hypernews.org/.

[LFK88] M. D. P. Leland, R. S. Fish, and R. E. Kraut. Collaborative document production using Quilt. In *CSCW '88*, pp. 206–215, Portland USA, September 1988. ACM.

[Nel87] Theodor Holm Nelson. *Literary Machines*. Mindful Press, 1987. (邦訳) リテラリーマシン, 1994, アスキー出版局.

[NKCM90] Christine M. Neuwirth, David S. Kaufer, Ravinder Chandhok, and James H. Morris. Issues in the Design of Computer Support for Co-authoring and Commenting. In *CSCW '90*, pp. 183–195. ACM, October 1990.

[Pre94] J. Preece. *Human-Computer Interaction*, chapter 28, pp. 588–590. ADDISON WESLEY, 1994.

[PSS94] François Pacull, Alain Sandoz, and André Schiper. Duplex: A Distrubuted Collaborative Editing Environment in Large Scale. In *CSCW '94*, pp. 165–173, Chapel Hill USA, October 1994. ACM.

[RWP95] Martin Rscheisen, Terry Winograd, and Andreas Paepcke. Content Rating and Other Thrid-Party Value-Added Applications for the World-Wide Web. *D-Lib Journal*, August 1995. http://www.cnri.reston.va.us/home/dlib/august95/stanford/08roscheisen.html.

[Sys] Sun Micro Systems. Java Servlets. http://www.javasoft.com/products/javaserver/servlets/index.html.

[VD95] Fabio Vitali and David G. Durand. Using versioning to support collaboration on the WWW. *the World Wide Web Journal*, Vol. 1, , 1995. http://www.cs.unibo.it/ fabio/bio/papers/1995/WWW95/version.html.

[vHP97] Arthur van Hoff and Jonathan Payne. Generic Diff Format Specification. Technical report, W3 Consortium, September 1997. http://www.w3.org/TR/NOTE-gdiff-19970901.html.

[佐藤 95] 佐藤円, 佐藤理史, 篠田陽一. 電子ニュースのダイジェスト自動生成. 情報処理学会論文誌, Vol. 35, No. 10, pp. 2371–2379, October 1995.

[石井 94a] 石井裕. グループウェアのデザイン, 第 3 章, pp. 18–20. 情報フロンティアシリーズ. 共立出版, February 1994.

[石井 94b] 石井裕. グループウェアのデザイン, 第 7 章, pp. 81–88. 情報フロンティアシリーズ. 共立出版, February 1994.

[門間 96] 門間信行, 大野浩之. ネットワーク上での協同執筆支援システム. 情報処理学会第 53 回 (平成 8 年後期) 全国大会 大会論文集 (3), pp. 383–384. 情報処理学会, September 1996. 講演番号 2J-10.

[門間 97a] 門間信行, 大野浩之. 協同執筆によるネットワーク上の人々が持つ情報の集積方法. グループウェア研究会研究報告. 社団法人 情報処理学会, September 1997. ISSN:0919-6072, 97-GW-24-2.

[門間 97b] 門間信行, 野田明生, 大野浩之. 小規模な組織の運営を支える情報共有機構−(2) 備品購入・管理システムの実装−. 情報処理学会第 55 回 (平成 9 年後期) 全国大会 大会論文集 (4), pp. 339–340. 情報処理学会, September 1997. 講演番号 3L-8.

[野田 97] 野田明生, 門間信行, 大野浩之. 小規模な組織の運営を支える情報共有機構 −(1) 備品購入・管理システムの設計−. 情報処理学会第 55 回 (平成 9 年後期) 全国大会 大会論文集 (4), pp. 337–338. 情報処理学会, September 1997. 講演番号 3L-7.

付録 A

拡張HTTPの詳細

A.1 HTTP プロトコル

rfc2068[FGM^{+}97] に示されているプロトコルの詳細は拡張BNF(augmented Backus-Naur Form) を用いて記述されており、本文でもこれにならう。まず、プロトコルのうち、以降の説明や実行例に関連する部分に関して抜粋しておく。

```
HTTP-message   = Request | Response

Request        = Request-Line
                 *( general-header
                  | request-header
                  | entity-header )
                 CRLF
                 [ message-body ]

Request-Line   = Method SP Request-URI SP HTTP-Version CRLF

Method         = "OPTIONS"
               | "GET"
               | "HEAD"
               | "POST"
               | "PUT"
               | "DELETE"
               | "TRACE"
               | extension-method

entity-header  = Allow
               | Content-Base
               | Content-Encoding
```

```
                       | Content-Language
                       | Content-Length
                       | Content-Location
                       | Content-MD5
                       | Content-Range
                       | Content-Type
                       | ETag
                       | Expires
                       | Last-Modified
                       | extension-header

        extension-header = message-header
```

拡張の結果、追加されたのはメソッドおよび、メソッドに固有のヘッダである。追加されたメソッドは上記の `Method` 中の `extension-method` に相当する。追加されたヘッダは上記の `entity-header` 中の `extension-header` に相当する。

A.2　ANCHOR メソッド

ANCHOR メソッドは、Request-URI で与えられた識別子で指定されるリソースに対して、部分参照のための新たな識別子を生成することを要求するためのメソッドである。

リソース中のどの部分を部分参照可能とするかは、Anchor-parameter ヘッダによって指定される。

```
Anchor-parameter = "Anchor-parameter" ":" #( 1*DIGIT )
```

Anchor-parameter ヘッダの示す内容の解釈は、リソースのメディアタイプによって異なる。リソースのメディアタイプが ”text/*” の場合、Anchor-parameter は 2 つの値 (x y) から成り、リソースの x 文字目から y 文字目までを指定する、と定める。

成功した場合、ステータス 201 (Created) を返し、Location ヘッダに新たに生成された識別子を格納して返す。失敗した場合は、その原因に応じて適当なステータスコードを返す。

例を示す。

```
<< ANCHOR /foo.txt HTTP/1.1
<< Host: aaa.bbb.ccc
<< Anchor-parameter 10, 20
```

成功した場合

```
>> HTTP/1.1 201 Created
>> Location: http://www.xxx.yyy/foo.txt#bar
```

失敗した場合

```
>> HTTP/1.1 405 Method Not Allowed
```

A.3　COMPOSE メソッド

COMPOSE メソッドは、

Compose-element ヘッダで指定される識別子で示されるリソース群を直列に並べて一括にし、Request-URI で与えられた識別子で指定される新たなリソースを生成するためのメソッドである。Request-URI で与えられた識別子がすでに存在する場合は、そのリソースの新たなバージョンとみなす。

```
Compose-element = #( compose-url )
compose-url     = "<" http_URL ">"
```

成功した場合、ステータス 200 (Ok) を返す。Request-URI とは別の識別子として生成した場合は、ステータス 201 (Created) を返し、Location ヘッダに新たに生成された識別子を格納して返す。失敗した場合は、その原因に応じて適当なステータスコードを返す。

例を示す。

```
<< COMPOSE /composed.txt HTTP/1.1
<< Host: aaa.bbb.ccc
<< Compose-element: <http://xxx.yyy.zzz/foo.html> \\
<<                  <http://kkk.lll.mmm/bar.html>
```

成功した場合

```
>> HTTP/1.1 200 Ok
```

失敗した場合

```
>> HTTP/1.1 405 Method Not Allowed
```