構造化電子文書のための流通経路管理機構

渡邊 諒† 大本 英徹‡

†京都産業大学大学院先端情報学研究科 〒603-8555 京都市北区上賀茂本山

‡京都産業大学コンピュータ理工学部 〒603-8555 京都市北区上賀茂本山

E-mail: †ryo4337@cc.kyoto-su.ac.jp, ‡oomoto@cse.kyoto-su.ac.jp

あらまし 近年，様々なソフトウエア，例えばワープロ・表計算などオフィスソフトウエアなどにおいて，そのデータ保存形式として XML が採用される事例が増えてきている．本研究では，XML 文書に加えられた編集作業情報（差分）を埋め込む事により，その履歴（編集内容，編集者）を管理する機構を提案する．本機構では，XML 文書の編集前後の差分をとり，元の XML 文書内に対して編集差分と編集者に関する情報を名前空間を用いて付加することで，編集履歴管理を可能とする．また任意のタイミングにて，編集履歴情報の第三者による認証が得られるサーバを用意することで，編集履歴情報の改竄がなく真正である旨を証明する事を可能とした．

キーワード 構造化文書, XML, 編集履歴, 流通経路

Design and Implementation of a Circulation Management Mechanism for Structured Documents

Ryo WATANABE† and Eitetsu Oomoto‡

†Division of Frontier Informatics, Kyoto Sangyo University Kamigamo-Motoyama, Kita-Ku, Kyoto 603-8555, Japan

‡Faculty of Computer Science and Engineering, Kyoto Sangyo University

E-mail: †ryo4337@cc.kyoto-su.ac.jp, ‡oomoto@cse.kyoto-su.ac.jp

Abstract XML is often adopted as the document format on a variety of recent modern software, for example, a word processor or a spread sheet. In this paper, we propose a mechanism to manage the editorial history of electronic documents which is in the form of XML. In our system, we extract the difference content between the original (pre-edited) document and the edited (after-edited) one by using DiffX. The difference content, and the editor's information (name, date, etc.) is encoded by XML name space, and is embedded into the edited document. With these embedded editorial history, we can trace the route on which the concerned document is circulated.

Furthermore, we have developed a server system for document authentication. With this server, we can certify any arbitrary document, which have some editorial history, that is, the circulation route.

Keyword Structured Document, XML, Editorial History, Circulation Route

1.はじめに

近年の XML フォーマットの普及は目覚ましく，電子商取引データのように異機種システム間で交換される電子データのみならず，図版(SVG)やワープロ・表計算といった所謂オフィスソフトにおけるオフィス文書保存形式としても用いられるようになってきている．ここでオフィスソフトを利用し

た一般的な文書の作成や利用を考えてみると，誰かが作成したワープロ文書がメール添付やネットワーク共有などを介して別の利用者に手渡され，そこで適宜加筆・修正された上で，他の利用者に渡されるといった事が日常的に行われている．

このような状況において，ある利用者が受けとった電子文書が複数人による加筆修正が加えられており，この文書「誰が，いつ，どのような修正・編集を加えてきたのか？」という「文書の出自」を明らかにしたい場合が起こり得ると考えられる．

ソフトウエアのソースコード管理では，CVS や Subversion などのバージョン管理システム（レポジトリ）を用いることで，世代管理や編集を行った利用者の特定を行う事は出来るが，逆にそれらレポジトリサーバの援用がない限り，編集履歴管理は困難である．従って，オフィス文書のようにアドホックに文書が作成され，メール添付や USB メモリや Web ダウンロードなど多種多様な経路にて流通して行くのが一般的な電子文書での編集履歴管理には，上手く適用出来ない．

本研究では，オフィス文書を含めた様々な電子文書形式として XML が採用されている事例が増えている事に注目する．XML では当該文書が本来表現する情報以外の付加的情報について，これを名前空間を用いて埋め込む事が出来る．そこで，文書に加えられた過去の編集内容やその編集を実施した編集主体に関わる情報を電子署名[1]を添えて埋め込む事により，当該文書自体にその出自，すなわち，文書の流通経路情報を保持させるという仕組みを提案する．

また，当該時点での電子文書に電子署名のみ付加する認証サーバを別途用意する事で，その文書の編集経路が真正であることを担保して，文書の信頼性を高める事を目指す．

以下，第２節では本研究の背景と基本的アイデアについて述べる．第３節では本研究で実装したシステム設計概略を示し，第４節ではシステム実行例を示す．第４節は結論である．

2.研究背景と基本的アイデア

2.1 電子文書の流通経路

近年では，コンピュータ技術やネットワークの発展により，電子文書の作成や配布が極く日常的に行われている．これはメール添付による転送や USB メモリを介した複製・配布など，紙媒体文書よりも電子文書の方が遥かに利便性が高いためである．このように様々な手段を通じて電子文書が一般的，商業的に扱われ，また転送や配布が容易なことから，ある１つの電子文書に対して複数人が関わりながら，順次，加筆修正してゆく作業は日常的に行われている．ここで我々は，１つの電子文書が複数人によって加筆修正されている場合，文書中の任意の箇所について「誰によって編集が行われたのか」，或いは「いつ編集が行われたのか」という編集履歴について注目する．

例として，あるグループ内における電子文書の流通を考える．図１はあるグループ内における電子文書の流通経路の例である．まず A が文書を作成する．その作成された文書を B が受け取り編集を加える．この時点では「追加，もしくは削除された内容の編集者は B」という編集履歴の追跡，管理は容易であると考える．しかし C や D に文書が渡り編集が行われ，さらに複数人に渡って編集を行われた場合「誰がいつ，どのような編集をしたのか」を断定することは難しく，編集履歴の追跡，管理は難しいと考える．

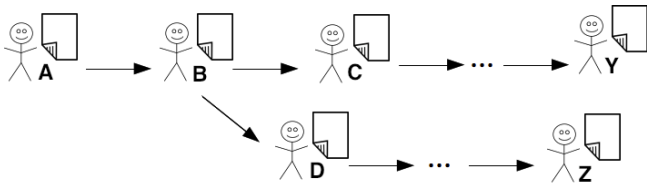

図１　文書の流通経路例

このような「誰によって編集が行われたのか」「いつ編集が行われたのか」という編集履歴の追跡管理は，例えば法的拘束力を有するような場合など，その文書の重要性や真正性が増すほど問題となる．このような背景により履歴管理，流通経路管理は重要であると我々は考える．

2.2 編集箇所の管理

流通経路を示す「誰が編集を行ったのか」「いつ編集を行ったのか」という情報の他に「どのような編集が行われたのか」という編集箇所の管理も必要であると考える．

編集箇所の管理の例を図２に示す．図２において，文書 a に対して編集が行われ文書 a'が作成された．この時文書 a に対して「山の左上に太陽を書き加える」という編集が行われている．

このように「どこに，どのような編集が行われたのか」という編集箇所の情報と共に流通経路の情報を管理することで，

「誰が，いつ，どのような編集を行ったのか」という文書の編集履歴を作成することが可能となる．

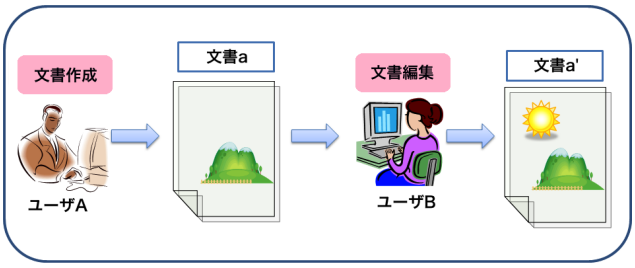

図２ 編集前後の電子文書の例

2.2 履歴管理

多様な流通経路を持つ電子文書において，編集履歴を追跡，管理するために必要な情報は編集内容，編集時刻，編集者 ID（編集者名）であると考える．これらの情報により「誰がいつ，どのように編集したか」を明確になり．この情報を管理できれば編集履歴を追跡管理することが可能となる．

本研究では，編集前文書と編集後文書を比較し，編集内容にあたる差分を取得する．またこの差分と編集者識別名，編集時刻に加えて，文書の復元に必要な情報を合わせ編集履歴として電子署名と共に編集後文書に埋め込む事で，編集履歴を文書単体で管理する事が可能となる．

2.3 XML 差分

２つの XML 文書を比較することで差分情報を取得する．例として図２の XML 文書を比較すると，以下の情報が得られる．

・要素 a の id 属性の削除
・要素 c の値”CCC”の削除
・要素 c の値”EEE”の追加
・要素 d の削除
・要素 f の追加

このように２つの文書を比較して得られた情報は，XML 文書の編集内容を示しており，この XML 差分を用いて編集履歴を作成する．

また XML 差分を取得するために，本研究ではオープンソースの Java API である DiffX [2]を用いた．DiffX は２つの XML 文書を比較し，文書内の要素，属性，値のそれぞれに対して追加削除の情報を表現した XML 文書を生成する．この DiffX を用いて図２の XML 文書を比較することにより得られる XML 文書を図３に示す．DiffX を用いて得られた XML 文書は <dfx:insert=”true”> などのタグにより，追加削除の情報を明確にして付加されており，この情報を元に差分情報を抽出し，編集履歴を作成する．

```
<root>
  <a id = "1234">AAA</a>
  <b>
    <c>CCC</c>
  </b>
  <d>DDD</d>
</root>
```

編集前のXML文書

```
<root>
  <a>AAA</a>
  <b>
    <c>EEE</c>
  </b>
  <f>FFF</f>
</root>
```

編集後のXML文書

図３ 編集前後の XML 文書の例

```
<?xml version="1.0" encoding="utf-8"?>
<root xmlns:dfx="http://www.topologi.com/2005/Diff-X"
xmlns:del="http://www.topologi.com/2005/Diff-X/Delete"
xmlns:ins="http://www.topologi.com/2005/Diff-X/Insert">
 <a del:id="1234">AAA</a>
 <b>
  <c><dfx:ins>EEE</dfx:ins><dfx:del>CCC</dfx:del></c>
 </b>
 <f dfx:insert="true"><dfx:ins>FFF</dfx:ins></f>
 <d dfx:delete="true"><dfx:del>DDD</dfx:del></d>
</root>
```

図４ DiffX を用いて取得した XML 文書

2.4 XML 文書の取り扱い

XML [3][4]は自由度の高い記述が可能であり，テキストベースのデータフォーマットであるため，異なるシステム間でのデータの共有が容易である．これにより近年，XML は企業間でのデータ交換フォーマットとしてなど様々な場面で利用されることが多くなってきている．

本研究の対象は，この XML 形式で表現された電子文書の流通経路管理にある．

2.5 OpenOffice.org アドオンとしてのプロトタイプ実装

XML 形式文書の取り扱いにおいて，XML 文書自体をそのままテキストエディタで加筆修正を行う事は殆ど無く，一般的には XML 形式で表現された特定目的向け文書を専用アプリケーションで操作する事が殆どであろう．本研究では，そのようなアプリケーションの極めて一般的なものとしてオープンソース統合ソフトウエア OpenOffice.org を想定し，このファイル形式である ODF 文書 [5]を主な対象とする．

また，OpenOffice.org は Java 言語などを用いたアドオンの形で機能拡張 [6]が出来るようになっているが，この機能を

を利用して，本研究では ODF 文書に対して編集履歴の埋め込み付加を実現する．これにより，エンドユーザに親しみ深い一般的アプリケーション環境において本研究の有用性を示す事が出来ると考える．

3.提案システムの構成と設計

3.1 システム全体の構成

図４はシステム全体の構成図である．本研究のシステムは編集者情報管理システム，履歴管理システム，XML 文書認証システムに分けることができる．編集者情報管理システムと履歴管理システムは OpenOffice.org へアドオンとして実装し，XML 文書認証システムはネットワーク上に独立して存在するサーバとした．

システム全体の流れは次のようになる．編集者情報管理システムに登録された情報を元に，履歴管理システムが「誰が，いつ，どのような編集を行ったのか」という編集履歴を XML 文書に埋め込み，履歴を含んだ XML 文書を作成する．また任意のタイミングで XML 文書認証システムへ作成された XML 文書を送信し，XML 文書認証システムは XML 文書を認証し，XML 文書に対して電子署名を付加して返す．

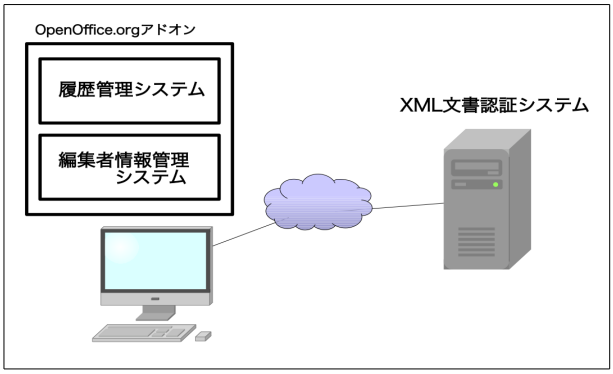

図４　システム全体の構成

以下，各システムについて説明する．

3.2 編集者情報管理システム

編集者情報管理システムは編集者情報の登録をサポートするためのシステムである． 本研究では，編集履歴管理に必要である『編集者名』と『電子署名付与のための鍵ペア』は，編集者本人が事前に準備する必要がある．また『編集者名』を「鍵生成時に入力する『姓名』と『エイリアス（別名）』」とを結びつけ，Java の keytool を用いてキーストア，鍵ペアの生成管理を行う．

この時 Java の keytool はコマンドプロンプトを使用して鍵ペアを作成するなど，一般的に扱いやすいとは言えないと考える．よって独自の入力フォームを用意し，必要項目を入力することで，容易に鍵ペアの生成管理がを行い，同時に編集者名の登録管理が行えるユーザインターフェイスを用意した．

また編集者情報管理システムは新規に鍵の生成を行い編集者情報を登録する『新規登録処理』，既存の鍵ペアを使用して編集者情報を登録する『再登録処理』を実装している．

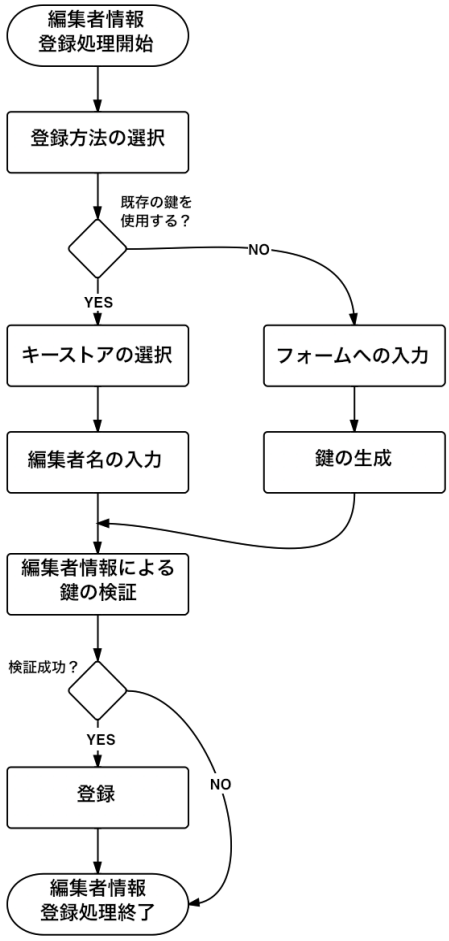

図５ 編集者情報登録処理

図５は編集者情報登録処理のフローチャートである．このフローチャートを用いて編集者情報登録処理の説明を行う．

まず最初に新規登録処理を行うか，再登録処理を行うか選択し，実行する．新規登録処理の場合，独自の入力フォームに必要事項を入力し，鍵を生成する．この時入力された『姓名』が編集者名として扱われる．再登録処理の場合，既存の鍵を使用するため，キーストアを選択する．また鍵を生成した際の『姓名』を入力する．

次に両処理の場合でも鍵の検証を行う．編集者名と鍵に用

いられる姓名，エイリアスとに相違がないかを検証することで，履歴管理システムにおいて使用できる鍵であるかを判断する．鍵の検証が成功した場合，編集者情報を登録し，登録処理を終了する．鍵の検証が失敗した場合，編集者情報を登録せず，登録処理を終了する．

3.3 履歴管理システム

編集履歴を作成管理する保存処理，また編集前の状態に文書を段階的に復元する復元処理を行うシステムである．

3.3.1 保存処理

図５は保存処理のフローチャートである．このフローチャートを用いて保存処理の説明を行う．

編集された文書を保存する際，編集前の文書と編集後の文書の比較を行い「追加された要素，属性，値」「削除された要素，属性，値」を差分として取得する．この差分に対して，編集者情報管理システムから提供された情報と復元の際に必要となる情報（表１）を属性として付加し編集履歴を作成する．また作成された編集履歴に対して編集者情報管理システムから提供された編集者の鍵を用いて電子署名を付与する．こうして得られた編集履歴を編集された文書に名前空間を用いて埋め込んで記録される．

属性名	属性値の説明
name	要素名
attributes	属性と属性値
value	値
count	親要素内の子番号
parent	親要素のフルパス
parentName	親要素の要素名
parentAttribute	親要素の属性名
parentValue	親要素の値
parentCount	親要素の親要素内での子番号
parentParent	親要素の親要素のフルパス
userName	編集者名
date	編集日時

表１　復元に用いる情報

3.3.2 復元処理

図６は復元処理のフローチャートである．このフローチャートを用いて復元処理の説明を行う．

復元を行う際，文書内の編集履歴を読み込み，編集履歴内の電子署名の検証を行う．これは編集履歴に対して偽造・改竄が行われていないかの検証を目的としている．検証が成功した場合,復元を許可する．また「どの段階までの復元を行うのか」ということを明確にするため，GUI を用いて文書内の編集履歴を一覧で表示する．この表示より編集履歴を選択し，その編集履歴を用いた段階的に復元を行う．

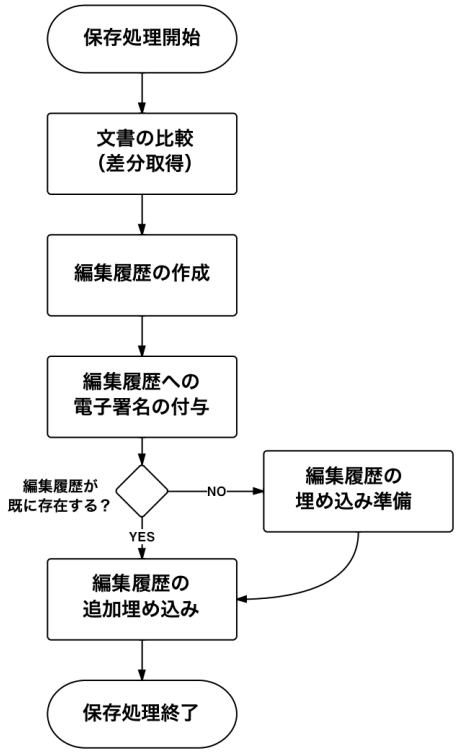

図５　保存処理のフローチャート

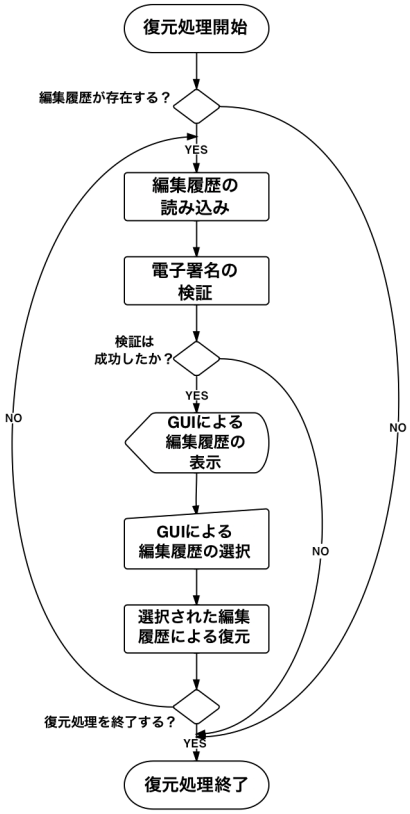

図６　復元処理のフローチャート

3.4 XML 文書認証システム

受け取った XML 文書に対して認証を行うシステムである．図 7 は認証処理のフローチャートである．このフローチャートを用いて認証処理の説明を行う．XML 文書を待ち受け，受け取った文書から編集履歴を読み込む．編集履歴内の電子署名の検証を行い，偽造・改竄が無い事を確かめる．検証が成功した場合，編集履歴として認証システムの電子署名が埋め込まれ，この署名が流通経路に改竄が無い事を証明する．署名された XML 文書は履歴管理システムに返送される．

この XML 文書認証システムで認証された文書の編集履歴には図 8 のような履歴が追加される．このように文書の編集履歴に名前空間を用いて signature タグが追加されており，また signature タグには XML 文書認証システムによる電子署名が付与されている．

signature タグは XML 文書認証システムの固有のタグとして用意し，認証された文書であると判別するためのタグである．

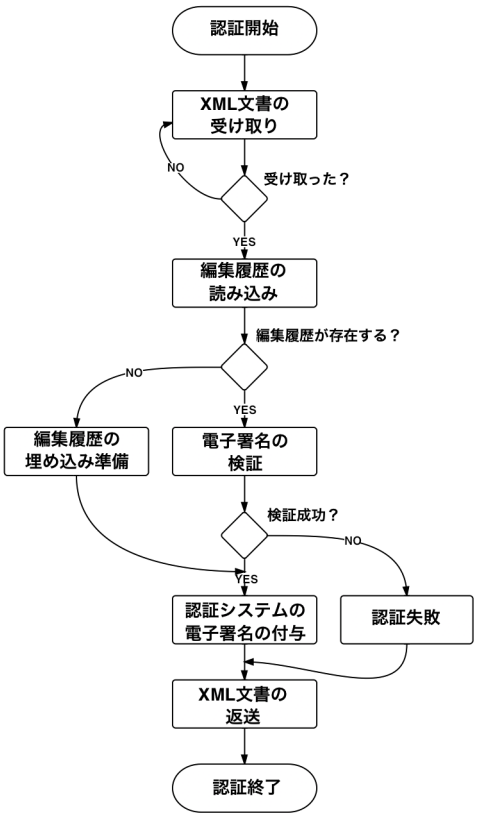

図 7　XML 文書の認証のフローチャート

```
<texthistory:signature userName="Authentication" date="2013_01_15_03:05_11"/>
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  - <SignedInfo>
        <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
        <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-s
```

図 8　signature タグ

4. システム実行例

本研究で作成したシステムの実行例を示す．

4.1 編集者情報管理システム

図 9 は独自の入力フォームである．新規に鍵ペアを作成するには，拡張されたメニューより『新規登録』を選択する．表示された入力フォームに必要事項を入力し，編集者名の登録と鍵ペアの生成管理を行う．

また既存の鍵ペアを使用するためには，拡張されたメニューより『再登録』を選択する．図１０に示すように，使用する鍵ペアが管理されているキーストアを選択し，鍵ペアを使用するために必要な情報を入力することで，編集者名と鍵ペアの情報を管理する．

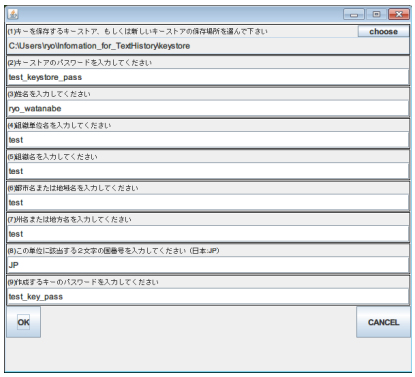

図 9　新規登録例

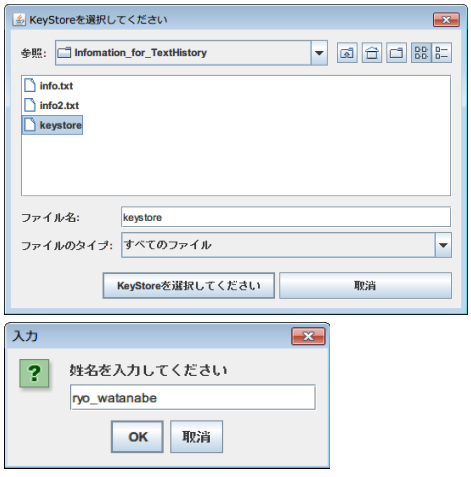

図１０　再登録例

4.2 履歴管理システム

図１１は編集の流れの例である．文書を新規作成する際は，拡張されたメニューより『新規保存』を選択し保存を行う．これにより編集履歴が付加された文書が新規作成される．また追加で編集・保存を行う場合は，拡張されたメニューより『上書き保存』を選択し，履歴を追加・保存する．

この保存処理によって作成された履歴を確認するためには，拡張されたメニューより『復元』を選択し，図１２のようなGUIが表示される．GUIには編集履歴の情報が一覧表示されており，編集者と編集日時を確認できる．またGUIはボタンで作成されており，押されたボタンに対応した編集履歴まで文書を復元し，編集内容の確認を行うことができる．ある編集履歴まで復元された文書の例は図１３である．

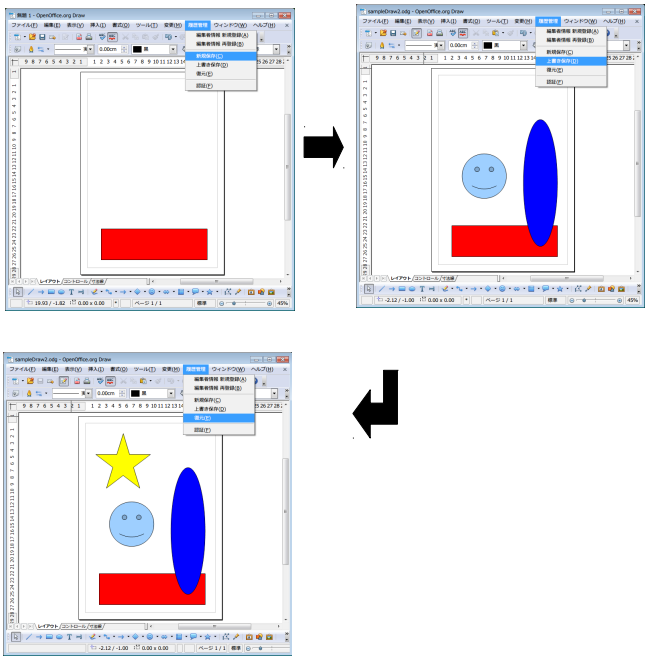

図１１　編集の流れの例

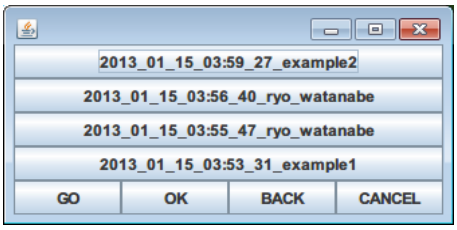

図１２　編集履歴情報の一覧表示

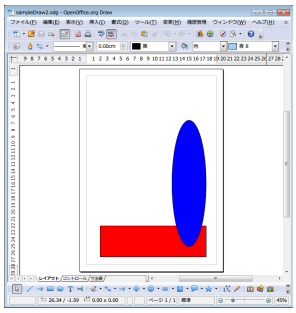

図１３　復元された文書の例

4.3 XML 文書認証システム

図１４は実際にXML文書を受け取ったXML文書認証システムの処理を示したコンソール画面出力の例である．

XML文書認証システムはXML文書を受け取り，XML文書内の編集履歴の検証を行う．検証が成功すれば，電子署名を付与し，返送処理を行う．

```
run:
受付開始.
受け取り完了.

認証を開始します
検証開始
検証終了

署名付与処理開始
署名付与処理完了

認証を終了します

返送処理
返送終了

構築成功（合計時間：４秒）
```

図１４　XML文書認証システムのターミナル画面の例

図１５はこのXML文書認証システムによって認証されたXML文書を示したものである．編集履歴を示すhistoryrootタグ内にsignatureタグが存在することが確認できる．またそのsignatureタグに対して電子署名が付与されていることも確認できる．

```
        </draw:page>
    </office:drawing>
</office:body>
<texthistory:historyroot xmlns:texthistory="http://texthistory.com/ns/texthistory">
  - <texthistory:history userName="Authentication" id="2013_01_15_03:05_11_Auth
        <texthistory:signature userName="Authentication" date="2013_01_15_03:05_
      - <Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
          - <SignedInfo>
                <CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml
                <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
            - <Reference URI="">
```

図１５　認証された文書例

6. 関連研究

ネットワークを介した分散協同作業に関する研究事例としては,中口らによる WhiteDog System [7] や SOBA プロジェクトによる SOBA フレームワーク[8] がある.これらシステムの目標は,メモリ上 のオブジェクトを共有可能な分散アプリケーション開発のためのプラットフォームを提供することにある．それに対し，本研究で扱う分散協同作業は「XML を基盤とする電子文書に対する追加編集作業」である．また，そのデータは極く一般的なストレージ媒体（ファイルシステム）に保存されている事を前提としている．また，本研究では一つのデータ（文書）の各箇所の編集に関わった作業主体を特定・追跡出来る事に主眼を置いており，この点でも方向性が異なっている．

さらに，Google 社が運用する商用サービス Google Docs [10] ではワープロ文書などの編集履歴管理機能を提供してはいるが, あくまでもサーバ上に保存された共有文書に対する機能であり，一旦，オフラインに保存された文書にまで適用されるものではなく，汎用的な XML 文書を対象ともしていない．

また，神戸大学の木俵らは,画像データに版権管理機構を Java オブジェクトとしてカプセル化する 行う方式を提案している[9].しかし,それはデータの不正利用防止を主な対象としており,電子文書の流通過程の管理を取り扱うものではない．

5. 結論

近年，XML をデータ交換フォーマットとして採用する事例が多くなり，電子商取引データやオフィス文書のファイル形式にも用いられるようになった．これらの電子文書は USB メモリ配布やメール添付など日常的に複製・転送・配布が行われている．

本研究では，XML 形式電子文書の差分を用いて編集履歴を生成し，その編集履歴を名前空間を用いて電子文書に埋め込む事で，文書データ単独での編集履歴と流通経路の管理を行うモデルを提案した．また，本手法の有効性を直観的に示すためのサンプルアプリケーションとして，OpenOffice.org へのアドオンとしてシステム実装を行った．また別途用意した認証システムによって電子文書の流通経路管理に真正性を持たせ，信頼性を高めることを図った．

今後の課題として以下が挙げられる．現在の実装では署名に必要な鍵ペアの生成・管理が公開鍵基盤(PKI)が想定する方式に十分に従っておらず，文書の真正性が完全には担保できていない．また，本研究で想定している文書編集プロセスは編集者から編集者へ文書が順次流通してゆくと想定しているが，オンライン上で共有状態にある文書の編集は全く考慮されておらず，その検討も必要であろう．

参 考 文 献

1. Mohan Atreya, Benjamin Hammond, Stephan Pain, Paul Starrett, Stephan Wu,『デジタル署名―技術, 標準と法的問題』, 翔泳社, 2003 年

2. DiffX API documentation, http://www.topologi.com/diffx/javadoc/index.html

3. Extensible Markup Language (XML) 1.0 (Fifth Edition), http://www.w3.org/TR/REC-xml/

4. Elliotte Rusty Harold, W. Scott Means,「XML クイックリファレンス」, オライリー・ジャパン, 2002 年

5. OASIS Technical Committee, https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office

6. OpenOffice.org for Developers, http://www.openoffice.org/development/index.html

7. The WhiteDog Project, http://www.ipa.go.jp/SPC/report/02fy-pro/report/882/paper.pdf

8. SOBA プロジェクト, http://www.soba-project.com

9. Yutaka Kidawara, Katsumi Tanaka, and Kuniaki Uehara, Encapsulating Multimedia Contents and A Copyright Protection Mechanism into Distributed Objects, Database and Expert Systems Applications, Lecture Notes in Computer Science Volume 1308, 1997, pp 293-302

10. Google Docs, http://docs.google.com