デジタル・ドキュメント 36－3
（２００２．１１．２９）

表示スタイル変更可能なWebトップXMLエディタ

矢尾　浩[†]　外山春彦[†]
白井　智[†]　金井達徳[†]

ユーザがXMLデータを扱う際に表示スタイルや機能を容易にカスタマイズして，かつその表示上で「見たまま」編集を可能にするWebトップXMLエディタを開発した．そのキーとなる技術が，単なるHTML文書を出力するユーザ定義のXSLT自体を構造変換して，編集可能なHTML文書を出力する機能を持つXSLTに作り変える「メタXSLT」である．本エディタを用いることで，ユーザは好みの表示スタイルのみを考慮してXSLTを記述すれば良く，編集機能をプログラムする複雑な作業から開放される．さらに，エディタ内でXMLやXSLTを統一的に扱うことができるので，XML文書単体で編集するだけでなく，複数のXML文書の閲覧・編集を組み合わせたより便利な使い方を提案した．

Web Top XML Editor supporting Operations on User-Defined Styles

Hiroshi Yao,[†] Haruhiko Toyama,[†] Satoshi Shirai[†] and Tatsunori Kanai[†]

We have developed a Web Top XML Editor, by which the user can edit XML Documents on the views by the user-defined styles. Meta XSLT is the key technology of this feature. Meta XSLT converts an user-defined XSLT into a new XSLT which genarates an editable HTML instead of a simple HTML. The User does not need to program the complicated editing functions, and needs only to describe the XSLT considering the looks of the view. Furthermore, we show the effective usage of this editor to browse and/or edit multiple XMLs on the same screen.

1. はじめに

XML(eXtensible Markup Language)[1] はデータフォーマットとしてここ数年の間に非常に大きな注目を集めた．現在では，様々な分野のドキュメントのフォーマットだけでなく，WSDLやSOAPといった情報交換を行うためのサービスプロトコルまでがXMLをベースとして定義されており，既に商用サービスとして運用される段階にまで達している．一般ユーザにとっても，リレーショナルデータベースのように最初にスキーマを厳密に設計しなくても使用可能であり，かつ可読なテキスト形式という点で導入しやすいデータ形式である．今後は企業・個人を問わず，XMLデータを扱う機会が更に増えていくと考えられる．

我々は長期の目標として，企業・個人を問わず広い層のユーザに，ビジュアルな環境で意図通りの情報処理ができる環境を提供したいと考えている．今回はその最初のステップとして，表示のカスタマイズに高度なスキルを必要とせず，ユーザが好みの表示スタイルでXMLデータを閲覧・編集できるWebトップXMLエディタを開発した．本エディタの特徴は次のようなものである．

- 標準規格 (XML, XSLT[2], HTML, JavaScript) をサポートするブラウザ上で動作．
- ユーザが記述したXSLTによる変換後の表示で「見たまま」XMLデータを編集可能．

今回の開発において，特に「見たまま」で編集可能とする技術のコアとなるのがメタXSLT技術である．この技術は，ユーザが記述したXSLT文書に我々が提供するメタXSLTを適用することにより，ユーザが記述した単なる表示形式変換用のXSLT文書を，編集機能付加命令を追加したXSLT文書に変換する技術である．ユーザはXSLT文書を記述する際には表示スタイルのみを考慮すればよく，編集機能をプログラミングする必要がない．

† (株) 東芝 研究開発センター
Corporate Research and Development Center, Toshiba corporation

以降では，まず第 2 章で既存の XML アプリケーションについて我々が問題と考えている点を明らかにする．第 3 章で我々の Web トップ XML エディタの全体が把握できるように，システムおよび機能の概要を述べる．第 4 章では本エディタの技術的特徴であるメタ XSLT を用いて編集機能を追加する方法について説明する．第 5 章では本エディタの利用形態を紹介し，第 6 章でユーザおよび開発者の観点から考察を行う．

2. 既存の XML アプリケーションの問題点

XML 文書を閲覧するには何らかのアプリケーションが必要である．XML はテキスト表現による文法を規定しているので，テキストエディタを用いて，テキスト表現のソースをそのまま表示することも可能である．しかし，XML のテキスト表現のままではユーザにとって必要な情報を把握しにくい．データの構造やデータが利用される状況に適した表示スタイルに変更する方が，はるかに判り易い表示が可能である．

XML 文書を扱う方法としては，

- XML データのスキーマに合わせた専用アプリケーションを使用
- 汎用の情報管理システム上での専用 UI を設計
- 汎用の XML エディタを使用
- 汎用ブラウザで閲覧

が挙げられる．

実行速度や見栄え，用途に応じた使い易さなどの点では，スキーマに合わせた専用アプリケーションが優位である．しかし専用アプリケーションでは，そのアプリケーションが提供する機能および表示スタイルしか利用できない．ユーザは，自分が必要とする機能および自分の好みの表示スタイルに近いアプリケーションを選択するほかない．

様々なデータを扱うグループウェア等の情報管理システムにおいては，表示スタイルを含めて管理者が設計した UI を使用できるようにして汎用性を持たせている．しかし，表示スタイルのフォーマットが独自のものである場合が多く，異なるシステム間での表示スタイルの流用は難しい．

XML 文書の編集という点に限れば，現在は各社からいくつかの XML 文書専用のエディタが提供されている．これらのエディタは様々な XML データを編集する上で汎用性は高いが，その反面そのエディタが提供する表示スタイル（一般的なツリー形式や表形式）上でしか編集ができない．一方，XML データのスキーマに合わせた表示スタイルを持つエディタを開発するには，表示のみならず編集動作についてもプログラムする必要があり，プログラミングに精通している必要がある．さらに，個人の好みの表示スタイルは千差万別であり，それらに合わせたエディタ全てを各個人が最初から開発することはコストが大きく，現実的でない．

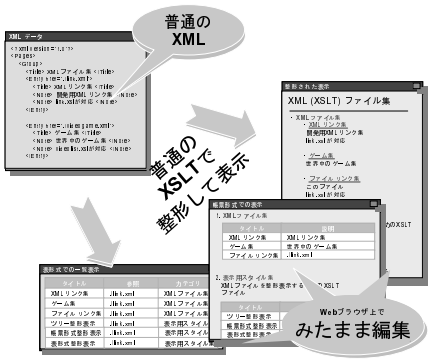

図 1 作業に合わせたスタイルでの編集
Fig. 1 Editting on the view preferable for a job

Web ブラウザで XML データを閲覧する手段としては，XSLT で表示スタイルを記述する方法がある．本来 XSLT は XML データの構造変換言語として規定されたものであるが，XML から HTML への変換規則を記述することによりスタイル変換に用いることができる．

特定のブラウザには HTML の編集モードを持つものも存在する．だからといって，XML データを XSLT で HTML に変換して，ブラウザ上で XML データを編集することができるわけではない．HTML エディタ上で編集可能な対象はあくまで変換後の HTML であり，変換前の XML データに編集結果が反映される機構はないからである．

結局，ユーザが簡単かつ思い通りに表示スタイルをカスタマイズして，その表示上で XML データを編集するには，現状では

- 既存の XML エディタでは，好みの表示スタイルに変更することができない．
- 表示スタイルに合わせた専用アプリケーションの開発にはコストがかかる．
- XSLT で HTML に変換しても，ブラウザ上で変換元の XML データを編集できない．

という壁が存在している．今回の Web トップ XML エディタの開発においては，この壁を取り払い，図 1 のようにユーザの作業に合わせたスタイルでのデータの編集・加工が容易に実現できる環境を提供することを目的とした．

3. Web トップ XML エディタの概要

3.1 システム構成

今回開発した Web トップ XML エディタのシステム構成を図 2 に示す．エディタを構成するプログラムは全て Web ブラウザ上で動作する．図 3 はブラウザ上のエディタの画面である．編集対象の XML 文書や HTML への変換を行う表示用 XSLT 文書等は外部のサーバに保存され，エディタは WebDAV プロトコルを用いてそれらの文書を転送する．

エディタとして我々が新規に開発したのは，

- ブラウザ上で動作する JavaScript プログラム群
- 編集機能の実現に必要な XML および XSLT 文書
- 初期画面用の HTML 文書

である．これらは Web ブラウザ側のローカルファイルシステムまたは任意の Web サーバ上に格納される．いずれの場合も初期画面用の HTML 文書をブラウザで閲覧することで，必要な JavaScript プログラム群，XML および XSLT 文書が読み込まれてエディタが起動する．OS, ブラウザ等の実行環境には既存のもの☆を使用する．

エディタを構成する JavaScript プログラム群はブラウザトップフレームワークとその上で動作する各種モジュール群に分類される．各モジュールは JavaScript のソースコードが埋め込まれた XML 文書の形態で外部に保管されており，必要に応じて動的にロード・アンロード可能である．

3.2 編 集 機 能

本エディタでは XML 文書の基本的な表示機能として，エディタ内に保持している XML 文書を表示スタイルで変換した結果の HTML 文書を描画する．表示スタイルには XML 文書を HTML 文書に変換する表示用 XSLT 文書を使用する．デフォルトの表示スタイルとして XML のテキスト形式で表示するスタイルが用意されている．ユーザ記述の XSLT を表示スタイルとして用いることもできる．

本エディタの最も特徴的な機能が，表示スタイルで変換した表示上で元 XML 文書を編集可能ということである．編集機能として以下の操作を用意した．

- フォーカスの指定
- テキストの編集
- 既存サブツリーの削除，移動，複製
- 既存サブツリーのカットアンドペースト

☆ OS に Windows2000/XP，ブラウザに InternetExplorer6, XML 処理エンジンに MSXML Parser3.0 を使用した環境で動作確認した．

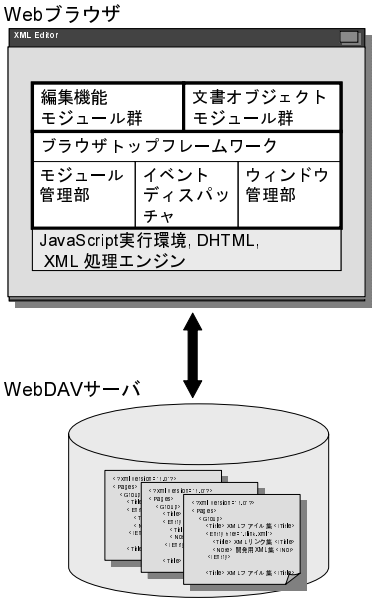

図 2 システム構成図
Fig. 2 Overview of Web Top XML Editor system

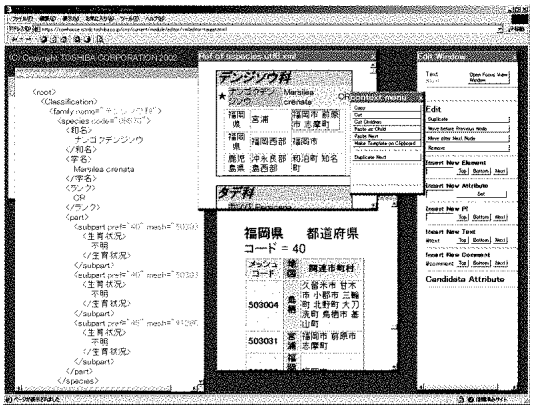

図 3 エディタの画面
Fig. 3 Screen of Web Top XML Editor

- 要素や属性等の新規ノードの追加

元 XML 文書を編集するには編集対象のノードを決定する必要がある．本エディタでは，元 XML 文書上での編集操作の基準位置をフォーカスとして管理する．図 4 に示すように，表示スタイルでの変換結果の HTML 要素は全て変換元の XML 文書のノードに対応付けられている．ユーザは画面上の HTML 要素をマウス等で指定することにより，対応する元 XML 文書上のノードをフォーカスとして指定する．

元 XML 文書の各種ノードの値を表示した部分はテキスト編集可能な HTML 要素で表示される．その部分にフォーカスを移動してテキストを編集すると，そ

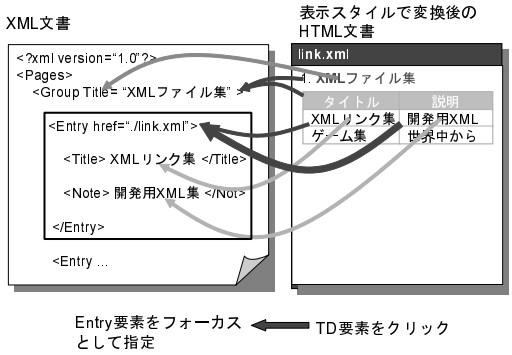

図 4 HTML を介したフォーカスの指定
Fig. 4 Focus selection through HTML

の結果が元 XML ノードの値にも反映される．

残りの 3 項目の編集機能は元 XML 文書の構造を編集する操作である．削除やコピーでは，フォーカスで指定された元 XML ノードを頂点とするサブツリーを対象とする．新規ノードの追加やペーストでは，フォーカスで元 XML ノードを挿入の基準位置として指定すると同時に，メニューで子供や兄弟といった挿入方向を指定する．

構造の編集の際には，元 XML 文書上での基準位置や挿入方向を指定するので，ユーザは元 XML 文書の構造についての知識を必要とする．

3.3 文書管理機能

我々が目指しているのは，単に XML 文書の編集を行うだけのエディタではない．ユーザが情報を整理したり，編集・加工するワークスペースとしてのツールを提供したいと考えている．本エディタではこれらの作業をサポートするべく，複数の XML 文書および XSLT 文書を容易に組み合わせる，あるいは同時に閲覧するための機能を用意した．代表的な機能を以下に挙げる．

- **マルチウィンドウによる複数 XML 文書の表示**
  図 3 で示すように，ブラウザ上の編集画面はマルチウィンドウ構成となっている．エディタ内の XML 文書毎にウィンドウを割り当て，同時に表示する．異なる XML 文書間でのカットアンドペーストが可能である．
- **表示スタイルを容易に切り替え**
  エディタ内には複数の表示用 XSLT 文書が保持される．それらの表示用 XSLT を選択できるメニューを用いて，ユーザは各 XML 文書の表示スタイルを切り替えることができる．
- **同一 XML 文書を複数ウィンドウで表示**
  ある XML 文書を参照して実体を共有する参照文書を作成できる．参照文書も含め，文書毎にウィンドウおよび表示スタイルの設定が独立しているので，XML 文書を同時に複数の異なるスタイルで表示可能である．
- **編集結果を即座に反映**
  XML 文書あるいは表示スタイルに設定された XSLT 文書が変更された時には再変換と再表示が行われる．

4. 表示スタイル変換後の編集

表示スタイル変換後の HTML 文書上で元 XML 文書を編集する手段として，ユーザが記述した XSLT 自体を変換して表示以外の機能を付加するメタ XSLT を開発した．本章では編集処理の流れと，メタ XSLT による編集機能の付加について説明する．

4.1 編集可能な HTML 文書

ユーザが記述した XSLT により出力した HTML 文書には元 XML 文書に編集操作を反映させるための情報も機構も存在しない．この節では，編集操作の起点となる編集可能な HTML 文書について説明する．

編集可能な HTML 文書としては，以下の要件を満たす HTML を出力する．

- 元 XML ノードの値に対応する HTML のテキストノードが画面上でテキスト編集可能
- HTML ノードから元 XML ノードへの対応関係が取得可能

ブラウザ上に表示される文字列（HTML のテキストノードの値）を編集可能にする方法はいくつか存在する．ブラウザに依存せず利用できる汎用的な方法は，編集可能にしたい文字列を TEXTAREA タグ等で囲む方法である．しかし，TEXTAREA の大きさや境界線等のスタイルを調整しなければ，メタ XSLT を使用しない場合に比べて大きく見た目が変わるおそれがある．

本エディタでは InternetExplorer の独自の機能である HTML 編集機能を使用する．HTML ノードに CONTENTEDITABLE 属性を付加することにより，HTML ノードの種類に応じてテキスト等の内容をユーザがブラウザ上でインタラクティブに編集できる．

また，本エディタでは，HTML ノードから元 XML ノードへの対応関係を取得するための情報として，対応する元 XML ノードの位置を表す XPath 情報を HTML 要素に独自の属性として付加する．この情報を元に，ブラウザ上での編集操作を反映させる元 XML 文書上での位置を決定する．

図 5 は本エディタで使用する編集可能 HTML の例である．(c) に示すように，元 XML ノードの値を表示したテキスト部分は SPAN タグで囲まれ，その SPAN

```
<books>
  <book>
    <title>枕草子<title>
    <author>清少納言</author>
  </book>
</books>
```

(a) 編集対象の XML 文書

```
<xsl:template match='books'>
  <TABLE>
    <xsl:apply-templates select='book'/>
  </TABLE>
</xsl:template>
<xsl:template match='book'>
  <TR>
    <TD><xsl:value-of select='title'></TD>
    <TD><xsl:value-of select='author'></TD>
  </TR>
</xsl:template>
```

(b) ユーザ記述の XSLT 文書

```
<TABLE xpath='1'>
  <TR xpath='1.1'>
    <TD xpath='1.1'>
      <SPAN xpath='1.1.1'
        <SPAN xpath='1.1.1.1'
          CONTENTEDITABLE='true'>枕草子</SPAN>
      </SPAN>
    </TD>
    <TD xpath='1.1'>
      <SPAN xpath='1.1.2'
        <SPAN xpath='1.1.2.1'
          CONTENTEDITABLE='true'>清少納言</SPAN>
      </SPAN>
    </TD>
  </TR>
</TABLE>
```

(c) 編集可能な HTML 文書

図 5 編集可能な HTML への変換
Fig. 5 Transformation to an editable HTML

タグに CONTENTEDITABLE 属性が付加される．また，全ての HTML 要素に元 XML ノードの位置を示す xpath 属性[☆]が埋め込まれる．属性の値には XPath の表記を独自に簡略化した形式を用いた．例えば 1 番目の TD 要素の xpath 属性の値は “1.1” であるが，これは

```
/node()[1]/node()[1]
```

を簡略化した記法である．これは図 5(a) の元 XML 文書の book 要素を指す．

4.2 編集操作の反映

図 6 は編集可能 HTML 上での編集操作を元 XML

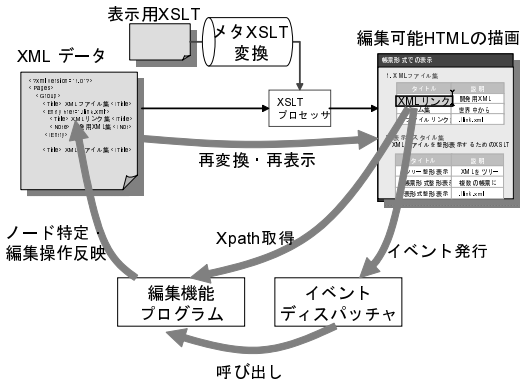

図 6 元 XML 文書を変更する流れ
Fig. 6 Flow of modifing the original XML Document

文書に反映する処理の流れである．編集可能 HTML 上での編集操作により UI イベントが発生すると，イベントディスパッチャに登録されたイベントマップを経由して，各編集機能を実装した JavaScript プログラムが呼び出される．

一般にブラウザ上でマウスクリック等の UI イベントを受け付けるには，対象となる HTML ノードにイベントハンドラを付加する．それに対して本エディタでは，イベントディスパッチャに UI イベントの種類と，条件と処理を記述した評価式の組 (イベントマップ) を事前に登録しておく．UI イベント発生時には種類の合致するイベントマップの評価式を評価して，条件に応じた処理を行う．

ユーザの操作を起点とする編集機能プログラムの多くは，イベントソースの HTML ノードに xpath 属性が付加されている場合に起動するようなイベントマップを，モジュールのロード時にイベントディスパッチャに登録している．従って，編集可能 HTML には xpath 属性を付加すれば良く，イベントハンドラを付加する必要がない．

起動された編集機能プログラム内では JavaScript の event オブジェクトの srcElement プロパティでイベントソースの HTML ノードを参照できる．フォーカスの移動操作の場合には，その HTML ノードから xpath 属性の値を取得してフォーカスを再設定する．その他の編集操作では，フォーカスから XPath 情報を取得して元 XML 文書を探索することにより元 XML ノードを特定する．その後は各編集操作に応じた処理を元 XML ノードに対して行う．

編集結果を元 XML 文書に反映すると，編集機能付加 XSLT による再変換を行い，出力された HTML 文書をウィンドウ内へ書き込む．

☆ 実際にはエディタ独自の名前空間に属するように，属性名にプリフィクスを付ける．

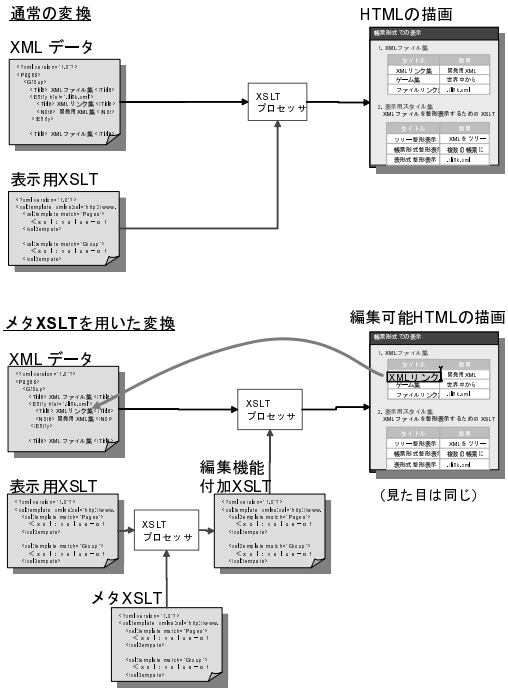

図 7 メタ XSLT による変換処理の流れ
Fig. 7 Flow of transformation with metaXSLT

4.3 メタ XSLT による機能付加

XSLT は XML の文法で記述されるので，XSLT 自体も他の XSLT で構造変換することができる．このように，XSLT を変換して新たな機能を持つ XSLT を生成する XSLT を，我々は「メタ XSLT」と呼んでいる．

XML から HTML への変換の流れの中でメタ XSLT が占める役割を示したのが図 7 である．通常は上図のように XML データをユーザ定義の XSLT で変換するだけである．メタ XSLT を用いる場合には下図のように，ユーザ定義の XSLT をメタ XSLT により変換し，機能を付加した一時的な XSLT を動的に生成して，ユーザ定義の XSLT に代わって XML データを変換する．

本エディタで用いるメタ XSLT は，ユーザが記述した単なる HTML 出力用の XSLT を，編集可能な HTML の出力を行う XSLT に変換するものである．この編集用メタ XSLT と図 7 の変換の流れを作る機構，および編集操作を元 XML データに反映させる機構をエディタが提供することにより，ユーザは編集機能についてのプログラミングから解放される．

4.4 編集用メタ XSLT

ユーザ定義の XSLT 内で何らかの HTML を出力する部分は，全て編集用メタ XSLT の変換対象となる．それらの変換対象を分類すると以下のようになる．

- XSLT 命令以外の要素を直接出力
- `<xsl:element>`命令により要素を直接出力
- `<xsl:copy>`命令により元 XML ノードをコピーして出力
- `<xsl:value-of>`命令で元 XML ノードの値を出力
- XSLT の default template による出力

編集用メタ XSLT ではこれらに対応するテンプレートを用意する．

簡単な例として，変換過程におけるカレントノードの値を表示する XSLT を変換する編集用メタ XSLT を考える．カレントノードの値を表示するには，次の XSLT 命令を使用する．

```
<xsl:value-of select='.'/>
```

この XSLT に，前節の編集可能 HTML を生成するように，xpath 属性および CONTENTEDITABLE 属性を付加する命令を追加する．

```
<SPAN CONTENTEDITABLE='true'>
  <xsl:attribute name='xpath'>
    <xsl:call-template name='xpath-string'/>
  </xsl:attribute>
  <xsl:value-of select='.'/>
</SPAN>
```

編集用メタ XSLT には上記の変換を行うテンプレートを用意する．以下にその例を示す☆．

```
<xsl:template
  match='xsl:value-of[@select='.']'>
  <SPAN CONTENTEDITABLE='true'>
    <axsl:attribute name='xpath'>
     <axsl:call-template name='xpath-string'/>
    </axsl:attribute>
    <axsl:value-of select='.'/>
  </SPAN>
</xsl:template>
```

本エディタでは，表示スタイル変換で出力される HTML 要素の全てに，出力時のカレントノードの位置情報を XPath 情報として付加する．上記では省略したが，`xpath-string` という名前のテンプレートは，カレントノードの位置情報を文字列として出力するテンプレートである．編集用メタ XSLT はこのテンプレートをユーザ定義の表示用 XSLT に 1 個追加する．カレントノードの位置情報は`<xsl:number>`命令を使用して出力する．

☆ プリフィクス `axsl` を持つ要素は，出力結果内では XSLT の名前空間に属する．この設定には`<xsl:namespace-alias>`命令を用いる．

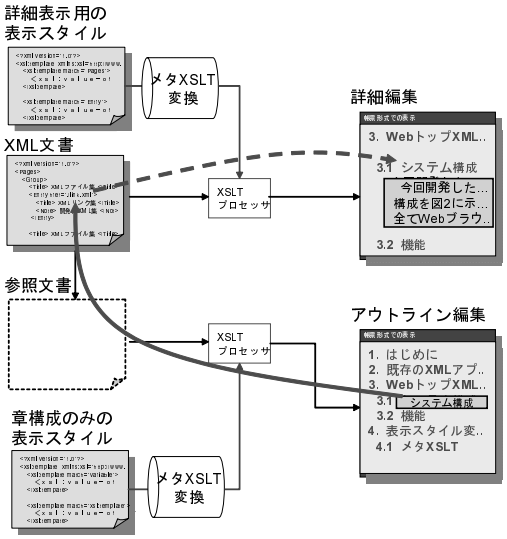

図 8 異なるスタイルでの同時編集
Fig. 8 Editing of a XML on multiple styles

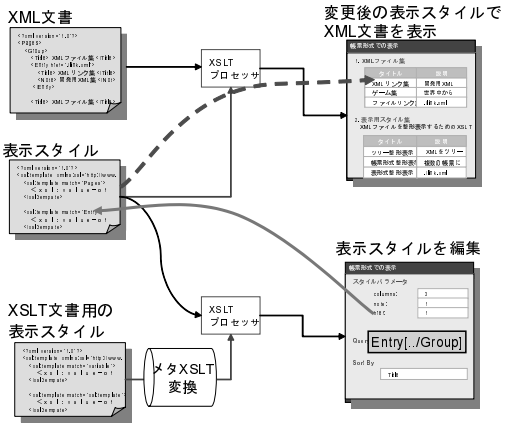

図 9 表示用 XSLT の編集
Fig. 9 Editing of a XSLT

5. 様々な利用形態

この章では，本エディタにおいて複数の XML 文書および XSLT 文書の表示・編集を組み合わせることで可能となる様々な利用形態の例を挙げる．

5.1 異なるスタイルでの同時編集

図 8 は参照文書を介して 1 個の XML 文書を 2 個のウィンドウで表示・編集する例である．一方には文書を詳細に表示するスタイル，もう一方には章題のみ表示するスタイルを設定する．一方のウィンドウで編集を行うと，もう一方のウィンドウでも即座に再表示が行われる．

章題表示のウィンドウ上では章構成を推敲する．章題を編集したり，元 XML 文書の構造の編集により各章の順序を入れ替える等のアウトライン編集が可能である．

5.2 XSLT の編集

XSLT 文書を編集する際には，複数のウィンドウを利用するとよい．図 9 に示すように，XSLT を編集するウィンドウだけでなく，その XSLT を表示スタイルにして別の XML 文書を表示しておく．これにより変換結果を即座に確認しながら表示用 XSLT の編集が可能である．XSLT や CSS，HTML の試行錯誤しながらの学習にも適している．

5.3 絞り込み編集

さらに，XSLT を編集するウィンドウにおける表示用 XSLT を工夫すると，入力フォームや編集メニューに似せたウィンドウを作ることができる．

XSLT では繰り返し処理を`<xsl:for-each>`命令等で記述する．これらの命令では処理対象となるノードを `select` 属性で指定する．そのパス式の条件部の記述に応じて，処理対象となるノードが絞り込まれる．そこで図 9 に示すように，`select` 属性のみを表示する XSLT を用意して，編集中の XSLT のウィンドウの表示用 XSLT に設定する．すると，絞込み条件（パス式）を入力するための条件入力フォームに似せたウィンドウとなる．

絞込み条件を変更すると XSLT 文書の `select` 属性が変更されて，その XSLT 文書を表示スタイルとする XML 文書も即座に再表示される．もちろん絞り込んだ表示上で元 XML 文書を編集することが可能である．元 XML 文書が非常に大きく目的の情報が見つけにくい場合には，このような利用形態が有効である．

6. 考　　察

6.1 ユーザから見た利点

これまでにも述べたように，最も大きな利点は，ユーザは表示のみを考慮して XSLT を記述すればよく，編集機能のプログラムを記述する必要はないことである．

また，特殊なツールで作成した XSLT 文書でなくとも元 XML データの編集が可能である．簡単な XSLT の記述はそれほど高いスキルを必要としないが，白紙の状態から目的の XSLT を作成することは手間がかかる．本エディタでは任意の XSLT を流用して，必要ならば自分の目的に合うように一部を修正すれば，その表示スタイル上での編集機能を利用できる．

例えば，本エディタには XSLT 文書を見やすく表示する表示スタイルが用意されているが，これは XML 文書をテキスト形式で表示するデフォルトの表示スタ

イルを流用して，テンプレートの単位を強調するようにカスタマイズしたものである．

Web ブラウザを編集画面のベースとして用いることで，エディタプログラムのインストールの必要がない．Web ブラウザは現在ほぼ全てのプラットホームに存在しており，Web サーバ側にエディタが格納されている場合には，そのサーバに接続できる環境であればどのクライアントマシンでもエディタを利用可能である．

6.2 開発上の利点

Web ブラウザを編集画面のベースとして用いることで，描画プログラムを作成する手間が省けた．

また，新たな機能を持つ変換プロセッサ自体を新規に作成するよりも，最低限必要な情報付加機能を持つメタ XSLT とリローダブルモジュールを記述する方が，はるかに容易に機能の切替えと拡張が可能である．

イベントディスパッチャにより，メタ XSLT を用いて HTML へ付加する情報と各種モジュールによる機能が切り分けられることも，機能拡張を容易にしている．メタ XSLT の開発時にはイベントハンドラの付加を考慮する必要はなく，モジュールの開発時にはメタ XSLT による情報付加の仕様のみを考慮すれば良い．

メタ XSLT は 1 種類に限らない．編集機能に関しても今回採用した方法以外に無数に存在するし，編集以外の機能を付加するメタ XSLT も考えられる．例えば，編集用メタ XSLT とほぼ同様の記述で，変換で使用された XSLT 命令の XPath 情報を付加するメタ XSLT が記述可能である．その情報から XSLT の変換過程を解析するモジュールを追加すれば，ユーザ定義の XSLT のデバッグのサポートに役立つ．

7. おわりに

本稿では Web トップ XML エディタの概要とその特徴的技術であるメタ XSLT について述べた．本エディタを利用することにより，ユーザは自分の作業に適した表示スタイルを選択し，その見た目のままで元 XML 文書を編集できる．ユーザは表示のみを考慮して XSLT を記述すればよく，編集機能のプログラムを記述する必要はない．

さらに XML 文書単体で編集するだけでなく，複数の XML 文書や XSLT 文書の閲覧・編集を組み合わせることで実現可能になる多様な利用方法を提案した．

今後の課題を以下に示す．

- 元 XML 文書の構造の編集のサポート
  構造を編集する際には元 XML 文書の構造についての知識がユーザに要求される．特に新規要素の追加の際には要素名および追加位置の指定が必要である．XML 文書，表示スタイル，スキーマなどから情報を抽出して，挿入可能な要素の候補を表示する等のサポートをすべきである．
- ブラウザ非依存化
  現状では InternetExplorer に依存する部分が多い．標準に沿ったプログラミングを進め，あるいはサーバサイドに処理を移すことにより，将来的にはブラウザ非依存にすることを目指す．

参 考 文 献

1) Tim Bray, et al: “Extensible Markup Language (XML) 1.0 (Second Edition)”, W3C Recommendation, http://www.w3.org/TR/REC-xml (2000).
2) James Clerk: “XSL Transformatins (XSLT) Version 1.0”, W3C Recommendation, http://www.w3.org/TR/xslt (1999).