文系論文のための XML と XSLT の作成

人文情報学科　0248060　常石龍一

——文系論文のための XML と XSLT の作成——

目　次

——文系論文のための XML と XSLT の作成——

1　序論

(1) テーマ

私の所属する大谷大学福田ゼミでは「人の役に立つデジタルツールの制作」が基本テーマとなっている。その中で私は「文系論文のための XML と XSLT の作成」を個人テーマに設定し、卒業制作及び卒業論文の執筆を行った。

卒業制作は、XML(eXtensible Markup Language) を用いて文系論文を作成する人を支援するためのデジタルツールを作成した。以下制作物を「EasyDoc」と呼称する。序論では、「EasyDoc」の概要と、誰の役に立つのか、どのように役に立つのか、ということを述べていく。

(2) EasyDoc の概要

まず、「EasyDoc」の概要を説明する。

「EasyDoc」は文系論作成の支援を目的としたドキュメント作成ツールである。XML を用い記述した文書ファイルを、XSLT(eXtensible Stylesheet Language Transformations) により HTML(HyperText Markup Language)、TEX[(1)]の 2 つのフォーマットに変換し出力することができる。構成は「xml2tex.xsl」と「xml2html.xsl」の 2 つのファイルとなっている。

(3) 対象とする人達

「EasyDoc」は次の方達の支援を目的として制作した。

1. 文系論文作成者
2. コンピュータに詳しくない人

3. 手間を省きたい人

では、それぞれにつていて説明していく。

(i) 文系論文作成者の支援

「EasyDoc」は一番目に挙げたように、文系論文に適したドキュメント作成支援ツールとして制作した。

文系に対象を絞った理由として、1 つは、私自身が文学部に所属しているということがある。身近に文系論文が豊富にあることで、文系論文がどのような形式で書かれているのか、そのためにソフトウェアにどのような機能が必要であるかということを知ることができる。また、私の技術的な点からも、ターゲットを絞ることにより、なるべく不具合のない物を制作できるのではないかと考えた。

もう 1 つの理由は、文系の人には理系の人ほどにコンピュータに関する技術がない場合が多い、ということである。理系の人であれば、アプリケーションのインストールや使い方などで多少の困難があっても自己解決できる知識や手段をもっていることが多い。また、自分に必要な機能を持ったソフトウェアをプログラムすることも可能である。これは、コンピュータに関する技術やソフトウェア開発などは工学、情報系といった理系の分野の仕事や、研究に携わっておられる人がほとんどであることから分かる。実際、「EasyDoc」を制作するきっかけとなった「SmartDoc」[(2)] の作者も理系の方である。

一方、文系論文作成者の場合、そのような人は少数である。中には、趣味でプログラムを学び、コンピュータ技術に詳しい人もおられるが、理系の方に比べ数が少ない。多くの文系の方は、コンピュータに関する知識も少なく、困難なことに直面すると解決できなかったり、解決に時間がかかったりする。このような場合、そのソフトウェアが便利なもの

であると分かっていても、使うことをあきらめてしまうのがほとんどではないだろうか。

そのような文系論文作成者も、手軽に使うことができるソフトウェアがあれば役に立つのではないかと考え「EasyDoc」を制作することにした。

(ii) コンピュータに詳しくない人

なぜ、コンピュータに詳しくない人の支援を考えるのかというと、既存のドキュメント作成ソフトウェアに簡単に使うことのできる物がなかったためである。既存のソフトウェアについては後述するが、役に立つということは、作業が楽になるということだけではなく、誰でもが簡単に使うことができるということも重要なのである。ここでの、コンピュータに詳しくない人というのは、全く知識のない人ではなく、ある程度コンピュータを利用している人を対象としている。

大学生は大半の人が卒業論文を書くことなる。文系、理系の学生を問わずにである。その多くが論文をコンピュータを利用して作成することになるだろう。事実、私の所属する人文情報学科では卒業論文は手書きではなくプリントしたものを提出しなければならない。

そこで、コンピュータを使い文章を書くことになるのだが、ほとんどの学生は「Word」(3) や「一太郎」(4) といったワープロソフトや、Microsoft Windows 付属の「メモ帳」といったテキストエディタを利用している。これらは手軽に使うことはできるのだが、途中で引用に関する注を書いたり、文章を挿入したりするとレイアウトがずれてしまい修正に労力を割かなければならくなることが多々ある。しかし、論文作成には TEX という優れた組版ソフトがあり、この TEX を使用すればそのような心配はない。

TEX とはスタンフォード大学の Donald E.Knuth 教授が活版印刷に劣らない美しい組版のできるコンピュータソフトウェアとして開発したものである。TEX は、上記のワープロソフトのような「WYSIWYG 方式」とは全く異なっている。

「WYSIWYG 方式」とは、「What You See Is What You Get（あなたが見るものはあなたが得るもの）」の略で、画面に表示されているものがそのまま印刷結果になるというものである。TEX では、それとは違いコマンドを挿入することで文書を作成していく。レイアウト等に関しては対応するコマンドを挿入しておけば自動的に処理されるため、論文の内容に集中することができる。

もともと、数式等を使った理系論文向けのソフトウェアであるが、文系論文にとっても以下のようなものが特長として挙げられる。

1. 非常に組版が綺麗である
2. TEX のファイルはエディタさえあれば書ける
3. 目次・脚注等の作成機能が優れている
4. フリーソフト(5)であり、誰でも入手できる

しかし、コンピュータに詳しくない人が TEX で文書を書くとなると決して簡単とはいえない。TEX のコマンドは非常に多くあり、どのコマンドが必要なのか、どのように使うのかということを理解するのは TEX を初めて使う場合やワープロに慣れ親しんでいる人にはとっつきにくい面がある。最初に述べたコンピュータに詳しくない人というのは困難なことがあると苦手意識を持ち、手を出しにくくなる傾向がある。そのような場合、TEX の便利さを理解する前に使うことをあきらめる人がほとんどではないだろうか。

そこで、TEX より知っている人が多く、作成が比較的容易である

HTML のような記述方法により書いた文書を、TEX 文書に変換することができるツールがあればそのような人達の役に立つことができるのではないかと考えた。

（iii）手間を省きたい人の支援

近年、ネット環境の整備が進んだことよりインターネットを利用する機会が多くなっている。本学もホームページは充実しており、学校紹介はもちろんのこと、履修している授業の時間割や、連絡事項等も閲覧することが可能となっている。これは、学問の分野においても無関係ではなく、例えば、大学での講義の内容をホームページで公開している教員の方も多数おられる。

そうなると、HTML 文書を作成する機会が多くなるのだが、その文書の多くは授業中に配布するプリントと同じことが多い。配布するものは、TEX やワープロソフトで、Web 公開するものは HTML で書くということは二度手間になってしまう。1 回だけというのならば問題ないが、授業のように何回もあるものだと面倒な作業となる。

実際、私の所属しているゼミの担当教員である福田教授も、「手作業では非常に面倒である。」とおっしゃっておられた。

「EasyDoc」を制作することにより、Web 公開と印刷を目的とした文書を頻繁に作成する際の手間を省くことができるのではないかと考えた。

（4）既存の物の問題点

「EasyDoc」を制作するにあたり、既存のドキュメント作成ソフトウェアである「SmartDoc」と本学の 2004 年度福田ゼミ卒業制作の「TEX2html」(6) について調べた。まず、「SmartDoc」について説明

する。

「SmartDoc」は「XML SmartDoc 公式リファレンスマニュアル」(p.3) によると、

「SmartDoc はドキュメント指向の XML フォーマットとこのフォーマットから各種ファイルを生成する処理系の総称です。」

とある。XML をベースとした SmartDoc 文書から HTML、TEX ファイル、プレインテキスト等を生成することができる。また、それぞれのフォーマットにも細かく対応しており、自由度も完成度も高いソフトウェアである。

しかし、技術文書やマニュアルの作成が主な対象となっているため、文系論文には必要のない機能が多い。また、処理に「Java」(7)を使用しているため実行には JavaRuntime(8)が必要となり、環境整備にも手間がかかる。これは理系の人やコンピュータに詳しい人にとってそう難しい作業ではない。しかし、先に述べた詳しくない人にとっては躓く要素となり得る。

次に、「TEX2html」であるが、これは TEX で記述された文書を HTML に変換するものである。プログラミング言語 Python(9)で記述されたプログラムであり、非常に手軽に TEX から HTML に変換することができる。しかし、全ての TEX のコマンドを HTML に変換することに対応するのは非常に難しいため、TEX2html では対応できるコマンドを制約し、不具合が出ないようにしている。それでも、実際に TEX 文書を変換するとエラーでうまくいかないことが多い。また、エラーが発生すると処理が中断されてしまう。

これは、TEX を元として変換しているためであり、最初からある決まった文書フォーマットを設計しそれに沿って文書を作成してもらえば

エラーの発生を極力少なく TEX、HTML の両方に変換することができるのではないかと考えた。

2　EasyDoc の詳細解説

(1) 制作方針

ここまで述べてきたことを踏まえ以下の点を考慮して「EasyDoc」を設計した。

1. 文系論文に必要十分な機能だけを実装しシンプルなものにする
2. 使えるをあらかじめ指定しておくことで変換の際の不具合を少なくする
3. HTML は文書構造のみ出力し、デザインは CSS で変更する
4. 環境整備も実行処理も簡単にできるようにする

1 と 2 については前述のとおりである。

3 につていは、文系論文を対象に考えているため、それほど複雑なデザインは必要ないのではあるが、全くのデフォルトのままでは見栄えがよくない。そのため、クラスを指定できるようにしておき、デザインに関しては「カスケーディングスタイルシート」(CSS) で利用者に指定してもらうことにした。HTML4.0 以降では「CSS」を用いてデザインを構成することが推奨されており、また、統一感のあるデザインにもなる。

4 は、先に述べた「SmartDoc」の環境整備の難しさということの解消をを目指した。「SmartDoc」を使用するためには JavaRuntime を入手しインストールし、次に「SmartDoc」を入手しインストールするという手順を踏まなければならない。また、実行に際しては、PATH(10) を通しておく必要がありコンピュータに詳しくない人にとっては難解な

作業である。

「EasyDoc」では Microsoft 社の「msxsl.exe」(11) を利用することにより同ファイルを作成した文書ファイルと「EasyDoc」のあるフォルダにおいておくだけで、インストール等の手間をかけずに変換を実行することができる。

「EasyDoc」の制作には XML を利用している。

XML は文書を構造化するためのマークアップ言語である。XML が考案される以前から SGML(Standard Generalized Markup Language) というマークアップ言語も存在していたが、インターネット上でそのまま使用できない、文書作成が難しく動作も重いといった欠点があった。

このような欠点を克服して生まれたのが XML である。SGML の基本的な考え方は継承しつつ、仕様の厳格さ、使用の難しさが解消されている。また、HTML のようにタグを省略したり、文書構造とデザイン要素を混在させるといったことはなくなっている。

この XML を利用することにより、次のようなメリットがある。

1. テキストエディタや XML エディタによる編集
2. 文書の永続性
3. データとしての価値

(i) テキストエディタや XML エディタによる編集

XML をベースとすることで、文書作成に既存のテキストエディタや XML エディタを活用することができる。これにより自身の使い慣れたエディタを使用することができる。

ワープロソフトはテキストの編集に加え、レイアウト処理を行うことを目的としているため処理が重たくなり、操作が複雑化してしまう。しかし、テキストエディタはテキストの入力と編集に特化しているため、

文書を書くという目的に非常に適している。

(ii) 文書の永続性

文書の作成には、作成した文書の有効期限が重要となってくる。ワープロソフトなどは独自のフォーマットを採用していることが多いため、その会社の都合によりソフトが使えなくなり文書に価値が無くなってしまう可能性がある。しかし、XML はオープンな技術であり、インターネットの世界での標準的なフォーマットになりつつある。これにより、文書としての価値は XML が使われている間、長期間保証される。

(iii) データとしての価値

「EasyDoc」は文書にレイアウト情報を含まず、純粋な文書の論理的な意味のみ記述する。そのため、データとしての価値も高い。

では、制作過程、構造を詳しく説明していく。

(2) 制作過程

まず、文系論文にはどのような機能が必要かということを知るため、十分な資料が必要である。そこで、図書館にある仏教学や史学についての論文、本学の過去の卒業論文を調べ、主に、見出し、注、段落、改行、太字やイタリック等の書体変更といったものがあれば十分なのではないかという結論に達した。理系論文のように数式等は必要なく、ビジュアル的にも凝ったものも無かったからだ。

しかし、先にも述べたように、Web 公開する場合、全くデザインのことを考えない、となるとあまりにも味気ない見栄えになるためクラスを指定できるようにして、CSS によりある程度のデザイン変更をできるようにした。

次に、変換に用いる XSLT の作成に取り掛かった。XSLT はプログ

ラム的な要素があり、なれるまで時間がかかった。ただ、要領をつかめば、XML 文書の要素 1 つずつに、対応している TEX のコマンドと HTML のタグを変換パターン（以下、テンプレートと言う）として作成しておけば変換することはできたので比較的簡単であった。

しかし、いくつか TEX と HTML の整合性を保つことが難しいところもあった。それについては XSLT の解説で述べる。

(3) XML の解説

XML 文書の基本構造について説明する。

XML にはまず XML 文書であるという宣言が必要なので、文書の一番上に、

「<?xml version=”1.0” encoding=”Shift_JIS” ?>」

という宣言文を書いておく。「encoding」は文字コード(12) を指定するもので「UTF-8」(13) を使用する場合以外は使用する文字コードを記述しておかなければならない。

そして、XML は唯一のルート要素が必要となるので、「EasyDoc」では「<root>」という名前の要素を使用することにした。XML においての要素とは <〜> で囲まれたものを指す。

要素には属性を記述することができ、宣言文の例でいえば「version」や「encoding」にあたる。「<root>」要素には「option」と「class」という属性を記述することができる。「<root>」要素は TEX に変換する場合は「\documentclass」として出力しており、オプションとドキュメントクラスをこの属性に記述する。

TEX と HTML ともに基本的な構造は、本文以外の設定等を書く部分と本文を書く部分、の 2 つで構成されている。このように、よく似て

いるため、XML の文書構造もそれに倣うことにした。おおまかな構造を以下に示す。

<?xml version=”1.0” encoding=”Shift_JIS” ?>
<root>
<head>
TEX のプレアンブル、HTML の head に相当
</head>
<body>
TEX の document、HTML の body に相当
</body>
</root>

TEX で使用するパッケージ等は「<head>」要素以下に「<pack>」という要素を用いて記述しておく。そして、「<body>」要素の中に本文を書き、文書を作成していく。

(4) XSLT の解説

XML はデータ記述言語であるため、そのデータをどのように出力するか、どのように人間の目に見せるか、といったスタイル・レイアウトに関する情報は一切含まれていない。そこで、XML 文書を他の文書形式に変換するのが XSLT である。1 個の XML 文書をさまざまな用途に利用することができるようになっており、この性質を「One Source, Multi Use」という。

今回、「EasyDoc」では XML 文書を変換するためにこの XSLT を利用することにした。これにより、変換処理に必要なものが「msxsl.exe」だけでよいため環境整備が簡単になる効果も期待できる。

XSLT は基本的にテンプレートの集まりで構成されている。XML 文書で使用する要素 1 つずつにテンプレートを記述する。例えば段落を意味する「 」という要素には、
TEX なら「<xsl:apply-templates />\par」
HTML なら、「 <xsl:apply-template /> 」
というテンプレートを記述しておく。これで、XML 文書に「 」要素で記述された部分を 1 つの段落としてそれぞれのフォーマットに変換出力することができる。

「<xsl:apply-templates>」は他に用意されたテンプレートを呼び出すための要素で、上の例であれば、「 」要素以下に他の要素が存在した場合、その部分は対応するテンプレートが呼び出され、処理される。これと同じように他の要素に対応するテンプレートを記述していくことで XSLT を作成していった。

多くの要素は上で述べたように単純に TEX と HTML に対応するコマンドとタグを出力すれば問題無かったのだが、いくつか工夫が必要であった要素もあった。以下、それらについて説明していく。

XSLT のソースコードを資料編として添付した。以降示す行数は、資料編のソースコードの行数を指している。

(i) 表組みの作成

TEX と HTML ともに表組みを出力することができる。文系論文作成者にとって表は、統計データの表示や、他のさまざまなデータを視覚的に分かりやすく示すために必要不可欠なものであり、レポートやレジュメにもよく使われる。

XML 文書での表組みのための記述は HTML とほぼ同じである。そのため、HTML に変換する場合は特に問題は無い。TEX は HTML と

は表を出力するための記述が異なるため、複雑なテンプレートとなった。

TEX で表を出力するには「\begin{center}〜\end{center}」で表の左右中央を指定し、その中に書く「\begin{tabular}〜\end{tabular}」で表自体を出力する。以下の行数は、資料「xml2tex.xsl」の行数である。

140 行目から左右中央の指定を、160 行目から表自体の出力を行っている。141 行目から 157 行目の「<xsl:choose>」は多岐分岐のための要素で、その下の「<xsl:when>」要素で条件にあてはまった場合の処理を、「<xsl:otherwise>」要素でその他の場合の処理を記述し、左右中央の指定の判定を行っている。

同様に 162 行目から 195 行目の間で行と表の終わりの判定を行っている。168 行目の「<xsl:for-each>」要素で「<td>」要素を繰り返し処理し、最後の「<td>」要素なら、つまり 1 行の終わりであれば「\\」を出力、そうでなければセルの区切りを示す「&」を出力するという処理を行っている。

表の終わりの判定は 165 行目の要素で行い、1 行の判定と同じように「<xsl:for-each>」要素で繰り返し、終わりであれば「\line」で行をはさんで出力、そうでなければ行の上部にだけ「\hline」を出力する、という処理を行っている。

(ii) 目次について

TEX は目次機能が充実しており、「\tableofcontents」というコマンドを記述すれば、自動的に目次が作成される。HTML にはこのような機能は無いため作成することにした。要素名は「<toc>」とした。ページ内リンクの機能も持たせている。なお、TEX では目次の部分のページ番号はローマ数字で、本文はアラビア数字で出力される。以下の行数は、資料「xml2html.xsl」の行数である。

まず、ページ内リンクを可能とするために、58 行目から 69 行目の間で、「<section>」要素を HTML の対応タグである、「<h1>」に変換する際に固有の数字を振ることにした。59 行目や 63 行目の「<xsl:element>」は、新たに要素を生成する要素で、ここでは、「<h1>」タグを生成している。同じように、61 行目や 63 行目の「<xsl:attribute>」要素で新たな属性を生成している。これらを組み合わせることで、XML 文書の「<section>」要素から、
<h1>1 序論 </h1>
というような HTML タグを生成することができる。

「name」属性の値がリンクがページ内リンクのための認識数字となっている。これは 64 行目の「<xsl:number>」要素により自動的に固有の数字が振られている。数字は「count」属性を指定することでカウントしている。

「<subsection>」や「<subsubsection>」も同様に処理している。また、「<xsl:number>」要素の「level」属性を「multiple」にすることにより、親要素[14]の認識数字を付加して生成するようにした。

目次の生成処理は 23 行目から 56 行目で行っている。上で説明したものと同様に「<xsl:element>」と「<xsl:attribute>」、「<xsl:number>」を組み合わせることにより生成している。番号なし箇条書きの「<ul>」タグを使うことで、視覚的に文章の構造が分かるようにした。

(iii) 予約文字について

XML にはそのままでは記述できない文字がある。それらは予約文字といい、例えば「<」や「>」、「&」等がある。XML 文書に記述する場合には「<」や「>」、「&」といった実態参照として記述しなければならない。

TEX にもそのままでは記述できない文字があり、こちらは特殊文字という。「<」であれば「$<$」、「&」であれば「\&」と記述しなければならない。XML 文書を HTML に変換する際には問題ないのであるが、TEX に変換する場合、このように予約文字と、特殊文字の記述方法が違うため問題となる。

解決方法として、予約文字を要素とすることにした。「<」と書きたければ「<lt />」と、「&」なら「<and />」というように書くのである。こうすることにより、予約文字の要素のテンプレートを作成しておけば TEX の特殊文字に変換して出力することができる。

(iv) 注について

論文なら理系、文系問わず必ず注が必要となる。もちろん、TEX にも注をつける機能はある。しかし、そのためにはコマンドをいくつか入力しなければならないため面倒であり、また XSLT で対応することも難しい。そこで、2004 年度福田ゼミ卒業制作物である TEX スタイル・ファイル「EasyLayout」(15)を利用することにした。そのため、注をつける場合には「EasyLayout」をプリアンブルに記述しておく必要がある。HTML には注の機能が無いため、注をつけた箇所の後ろに「(注内容)」と出力するようにした。

注をつけたい箇所に <footnote> という要素で注内容を記述しておき、注を打ち出したいところに <outnote> 要素を記述すれば注を出力することができる。但し、後注としてのみ対応している。

3　EasyDoc の評価

(1) アンケート結果と考察

今回制作した「EasyDoc」を福田ゼミ三年生の方に実際に使い、アンケートに答えていただいた。5 人の方から回答を頂くことができた。アンケートをとる目的は客観的に「EasyDoc」を評価してもらうところにある。人の役に立つツールの作成というテーマに沿って制作しているため、私一人だけの考えでは完成しない。必要だと考えた機能も他人には必要ないと感じられるかもしれないし、逆に簡潔にしたつもりのところが必要だと感じられるかもしれない。デジタルツールを制作していく上で実際に使ってもらいフィードバックすることは非常に重要且つ有効なのである。

それでは、アンケート結果について考察していく。

まず、一番重要な「EasyDoc」は必要かどうか、ということについての回答は、5 人の方全員から必要であるという答えを得ることができた。これは今回のテーマで、一番大切なことだ。自分だけが使うのだけではなく誰かに必要とされなければ人の役に立っているとはいえないからである。

次に、制作する際に心がけていたなるべく簡単にシンプルにという点についてであるが、これにも良い評価をいただくことができた。アンケートを実施した時間は短かったためすべての要素を使用するような文書を作成することはできなかったが、逆に短い時間である程度理解してもらえたということは当初の目標を達成できたものであると考える。ここまでツール自体の評価はおおむね良い評価を得ることができた。

しかし、5 人の方に共通して改善したほうがよいと指摘された点が

あった。作成したファイルを HTML 又は TEX に変換する際の方法についてである。原因はアンケートの際のマニュアルにあった。実行方法だけを書き、エラーが出た場合の対処方法等が十分ではなかったためだ。

「EasyDoc」の対象とするものにあるように、コンピュータに詳しくない人に問題なく使っていただくためには、使用方法の簡単さも当然ながら、マニュアルの重要性も再認識することができた。

(2) 自己評価

今回の卒業制作「EasyDoc」は文系論文作成者、コンピュータに詳しくない人、手間を省きたい人の支援、また従来の問題点について考慮して制作を進めてきた。それぞれについて結果を述べていく。

文系論文作成者、コンピュータに詳しくない人にも分かりやすいように、環境整備に関しては MicrosoftWindows が動作するのであれば作成したファイルと「msxsl.exe」、「EasyDoc」の 3 つを同じフォルダにおくだけで実行でるようにした。ソフトウェアのインストール作業が一切必要ない。このことは手間を省きたい人の支援にもつながる。

従来の問題点であった使用方法の難しさについては、文系論文に対象を絞ることにより、覚えることを少なく、簡単に使えることを目指した。この方向性は間違ってはいなかった。アンケート結果からも簡単に使うことができたとの回答を得ることができ、実際に、私自身の卒業論文を「EasyDoc」で書いたのだが特に不具合なく TEX、HTML の両方に簡単に変換することができた。しかし、まだ改善しなければならない点がいくつかある。

(i) 予約文字に関する処理

「<」や「>」といった予約文字の入力についての処理は先に述べたが、これが利用者に手間をかける結果となってしまった。XML と TEX の予約文字の違いを利用者が意識することなく処理することができなかった。また、予約文字を多く含む文章を書く場合、見づらくなってしまい、それが XML 文法エラーを引き起こす原因にもなっている。

今のところ、現状以上の解決方法は考えついていない。XSLT にもっと習熟すれば可能なのかもしれないし、XSLT を使用して変換処理を行うのは無理なのかもしれない。

(ii) tex の特殊文字についての処理

TEX にはさまざまな特殊文字がある。今回、TEX や LATEX については対応することができた。しかし、これも <tex /> や、<latex /> と入力してもらわなければならず、多少の手間がかかる。また、見出しのタイトルを要素の属性に記述するようにしたため、タイトルに予約文字や特殊文字を使うことができない。つまり、タイトルの中に < や TEX といった文字を使用することができないのである。これは、XML の属性には要素を記述することができないためである。

(iii) 参考文献の処理

論文は必ず参考にした文献を最後に記述する。TEX では参考文献について記述するためのコマンドがあるが、これに対応することは、私の技術的な問題でできなかった。そのため、動作は番号なし箇条書きと同じではあるが、参考文献という意味をもたせるため「<biblio>」という要素を用意することで代替とした。

（iv）DTD の作成

XML 文書のフォーマットを定める DTD(Document Type Definition)[16]も作成する予定であった。しかし、XSLT の作成に時間がかかり作成することができなかった。

（v）表の処理

TEX、HTML ともに多彩な表を作成することができる。しかし、今回はセルの合成等ができず、シンプルな表しか作成することができない。TEX では縦の罫線のみ一括ではあるが出力を指定できる。

上記のようにまだまだ改善が必要な箇所や、不足している機能があり、2005 年 12 月現在、まだとりあえず動作する物ができた状態だと考えている。実際に使ってみると、やはり予約文字や特殊文字を要素として記述するのは面倒だったし、XSLT の変換処理の記述にも冗長な箇所があると思われる。

アンケートは取ったが、もっと実際に使っていただき多くの人の意見を聞き、改良していかなければ、本当に文系論文に適した物は完成させることはできない。そのためには、マニュアルを充実させホームページを立ち上げての Web 公開も必要になってくるだろう。

4　結論

（1）まとめ

今回「人の役に立つデジタルツールの制作」をテーマに卒業制作を行ってきた。この、「人の役に立つものを作る」ということは非常に難しいことであった。なによりも、何が役に立つのか、何が必要とされているのかということを見極めるのが大変で、個人テーマの「文系論文の

ための XML と XSLT の作成」も私一人では思いつくことができずに、担当教員である福田教授からアドバイスを頂くことにより決めることができた。

また、技術的な問題も多々あった。「EasyDoc」の制作には XML、HTML、TEX の知識を必要とした。

XML については一から勉強を始めた。基本構造のところで述べたが、それぞれの要素に XSLT のテンプレートを作成し「<xsl:apply-templates>」要素で呼び出す、という方法を考えつくまでにかなりの時間がかかってしまった。

HTML、TEX に関しては初めてではなかったが、忘れていることが多くあり、制作と並行して勉強したため全体的に知識が足りなかった。結果として、先述した問題点や妥協することになった箇所ができてしまった。また、制作物が形となるのが遅くなったため多くの人から意見を聞き、改良することができなかった。

卒業論文及び卒業制作についての総括であるが、制作した「EasyDoc」は一応形にはなったものの、決して完成度の高いものとはいえない。実際の使用に堪え、多くの人が簡単に使うことができ、便利だと感じるレベルに達するには、まだまだ改良を重ねていく必要があるだろう。しかし、「EasyDoc」のようなソフトウェアの必要性はアンケートや、実際に使ってみることにより感じることができた。

最後に、この論文が、私自身の今後に生かすことのできるものであり、また「文系論文の作成の支援」という分野において新たなデジタルツールを作成するきっかけになることを願っている。

注

（1）http://oku.edu.mie-u.ac.jp/ okumura/texwiki/

（2）http://www.asahi-net.or.jp/ dp8t-asm/java/tools/SmartDoc/index_ja.html

（3）http://www.microsoft.com/japan/office/word/prodinfo/default.mspx

（4）http://www.ichitaro.com/2006/

（5）ユーザがソフトウェアを実行、複製、頒布、研究、変更、改良することができる

（6）http://python2004.hp.infoseek.co.jp/

（7）http://jp.sun.com/java/

（8）http://www.java.com/ja/download/manual.jsp

（9）http://www.python.jp/Zope/

（10）http://e-words.jp/w/E38391E382B9.html

（11）http://www.microsoft.com/downloads/details.aspx?FamilyID=2fb55371-c94e-4373-b0e9-db4816552e41&DisplayLang=en

（12）http://e-words.jp/w/E69687E5AD97E382B3E383BCE38389.html

（13）http://e-words.jp/w/UTF-8.html

（14）直接の上位要素

（15） http://tibet.que.ne.jp/otani/wiki/index.php?EasyLayout

（16）文書型宣言。XML 文書の構造を定義するもので他に「XML Schema」、「RELAX NG」等がある。

参考文献

・浅海智晴

2002　『XML SmartDoc 公式リファレンスマニュアル』ピアソン・エデュケーション

・奥村晴彦

2000　『改訂版 LaTeX 2_ε 美文書作成入門』技術評論社

・山田祥寛

2004　『基礎 XML』インプレス

・山田祥寛

2004　『XML 辞典』翔泳社