データベースシステム 128－29
（２００２．７．１８）

XML 操作スクリプト言語 xTrics の提案

岡本辰夫 [*1]　國島丈生 [*2]　横田一正 [*2]

[*1] 岡山県立大学大学院情報系工学研究科　[*2] 岡山県立大学 情報工学部
tatu@c.oka-pu.ac.jp　{kunishi,yokota}@c.oka-pu.ac.jp
〒 719-1197 総社市窪木 111　〒 719-1197 総社市窪木 111

近年，コンピュータやネットワークの発達に伴い，Web 上に様々なデータが大量に存在するようになってきた．このような Web 上のデータにおけるデータ記述言語として XML が注目されており，様々な分野で XML によるデータ記述の仕様が開発されてきている．しかし，1 つの分野でも様々なデータ記述の仕様が乱立し，互いの情報に整合性が取れない場合が増えてきたために，情報を同一の構造に変換することが重要となってきた．そのため本研究では，簡単な記述で XML 文書を操作することができ，文字列に対する強力な操作をもつ XML 操作スクリプト言語 xTrics を提案している．また既存の構造変換言語である XSLT と比較し，xTrics を有効に利用する分野について考察した．

キーワード: XML，スクリプト言語

A New Script Language “xTrics” for XML Transformation

Tatsuo Okamoto[*1] Takeo Kunishima[*2] Kazumasa Yokota[*2]

[*1] Okayama Prefectural University
Graduate School of System Engineering
tatu@c.oka-pu.ac.jp
111 Kuboki, Soja, Okayama 719-1197

[*2] Okayama Prefectural University
Faculty of Information Science and System Engineering
{kunishi,yokota}@c.oka-pu.ac.jp
111 Kuboki, Soja, Okayama 719-1197

XML has become a *de facto* standard for exchanging various kinds of data on the Web and many kinds of data have been written in XML-based languages such as SVG, MPEG7, and VoiceXML. However, as description policies depends on each user’s intentions, their descriptions are not neccesarily normalized even if XML Sclema is fixed. In this paper, we propose a new script language called xTrics for XML transformation, which is aimed not only for syntactic tansformation but also semantic one such as ontology differently from XSLT.

Key words: XML, script language, XML transformation

1 はじめに

近年，コンピュータやネットワークの発達に伴い，Web 上に様々なデータが大量に存在するようになってきた．このような Web 上のデータにおけるデータ記述言語として，XML が注目されている．XML は，半構造化データの記述言語として広く利用されており，電子化された情報の交換フォーマットとしてデファクトスタンダードとなっている．そして，XML の記述能力の高さから様々な分野でデータ記述を XML で行ない，Web を介して情報交換をするための技術開発が行なわれている．

しかし，1 つの分野で様々なデータ記述仕様が乱立し，互いの情報に整合性が取れない場合がある．そのため，ある XML 文書を他の XML 文書に変換する機能をもつ XSLT[3] を利用して，互いの情報を同一の構造に変換することがよく行われている．しかし，XSLT には次のような問題点がある．

1) 変換する文書の構造が複雑になればなるほど，その記述は加速度的に煩雑になる．
2) 文字列内に含まれる特定の単語やパターンを簡単に置換する機能はない．

ここで 1) の問題は，XSLT は宣言的にルールを記述するが，ルールをモジュール化することが出来ないため似たようなルールを何度も各必要があり，コードが長くなる．また 2) の問題は，例えば<名前><姓>岡本</姓> <名>辰夫</名></名前> という XML に対して，

```
<xsl:template match="名前">
 <名前><xsl:value-of select="姓"/>
 <xsl:value-of select="名"/></名前>
</xsl:template>
```

のような XSLT を実行すると<名前>岡本 辰夫</名前> が簡単に得られる．しかし逆に，<名前>岡本 辰夫</名前> から<名前><姓>岡本</姓> <名>辰夫</名></名前> を導出する場合は,「岡本 辰夫」という文字列を「岡本」と「辰夫」に分けなければならないため XSLT では困難である．これは特に，図 1 のような文書系 XML の構造を変換する際には致命的な問題点となりうる．

われわれはこれまで，マルチメディアコンテンツを利用者の要求に応じて提示するシステムについて研究してきた [1]．そのなかで，コンテンツの個人化

```
<section>
 <para>
  鬼ノ城は，663 年の白村江の戦いで唐・新
  羅の連合軍に大敗し，その連合軍が日本に
  侵攻する危機感から，国土防衛のため，北
  九州～瀬戸内沿岸～畿内にいたる西日本各
  地に築いた，古代山城の一つと考えられて
  います．しかし，<marker color="red">
  『日本書紀』などに築城などの記載はまっ
  たくなく</marker>，これまで謎の城とい
  われてきました．
 </para>
 <para>
  国指定史跡である鬼ノ城を将来的には整備
  をしていくため，いまだ不明の点が多いこ
  とからまず基礎データを得るための発掘調
  査を実施し，その成果をもとに整備計画を
  進めていく方針で，これまでに 3 回にわた
  る発掘調査を実施してきました．
 </para>
</section>
```

図 1: XML の例

に関する研究 [2] を行い，文書系 XML に対しての構造変換が必要となった．しかし，上記の問題点により XSLT の適用は困難であると考えた．そこで本研究においては，簡単な記述で XML 文書を操作することができ，文字列に対する強力な操作をもつ XML 操作スクリプト言語 xTrics(XML Tree Information Control Script) を提案する．

本稿の構成は次のとおりである．2 節で本研究において提案する XML 操作スクリプト言語 xTrics について述る．そして，3 節で既存の XML 操作技術との比較を行う．最後に 4 節では研究のまとめを述べる．

2 xTrics の概要

xTrics は，XML が持っている木構造の各ノードに対して，追加や削除，変更などの操作を行うことを目的として研究開発しているスクリプト言語である．xTrics は，Python をベースとして XML に対して強化した手続き型のプログラミング言語である．

その主な特徴として以下のことなどが挙げられる．

- XML の文章としての連続性の保証
- 木構造に対する操作
- 分岐や繰返しなどの制御フロー
- 文字列に対する強力な操作

ここで文章としての連続性保証とは，XML はもともと構造化文書の記述言語として作られているため，xTrics を用いて任意の一部分を切り出す操作 (後述するスライス操作の適用) を行ったとしても，明示的に順序を入れ替えない限り必ず元の文章の順序を保証することである．つまり xTrics においては，XML を木構造として捕らえながらも文章の連続性を保つために各ノードがシーケンシャルに一列に並べられるように考えている．

2.1 xTrics のデータ型

xTrics は，大きく数値型，文字型，シーケンス型の 3 つのデータ型をもつ．

数値型は 123 や 45.678 のように数字ので記述し，文字型は'a' のようにシングルクオートで記述する．シーケンス型は [a,b,c] のように角括弧とカンマで区切られた要素により記述する．

ここで文字列は，文字型をデータをして持つシーケンス型で表す．文字列は"abcd"のようにダブルクオートで記述され，['a','b','c','d'] のような文字シーケンスと同等の意味をもつ．

また，一般的に XML のデータモデルは，タグの入れ子構造を木として捉える．xTrics においても同様であるが，前述したように XML の文章としての連続性を考慮する必要がある．XML のひとつのタグに注目すると，その子供には必ずタグか文字列が来て，それらの要素間に明確な順序があるために シーケンスで表現できる．つまり，xTrics における XML は入れ子シーケンスであると考える．

しかし，このように XML を考えるとタグ名や属性の情報が欠落している．そこで XML を　　，Element@Att1="val1"…@Att$n$="val$n$" [value1,…,value$m$] のようにタグ名のあとに@で属性を列挙し，角括弧とカンマで区切られた要素で記述する．XML において [value1,…,value$m$] の部分を省略した場合は空要素タグとして扱う．

そのため，xTrics における XML のデータモデルはラベルつき木構造であると考えることができる．また，文章としてみた場合そのデータは行きがけ順にシーケンシャルに並んでいると考える (図 2 に参照されたい)．図 2 は図 1 の XML 文書のデータモデルである．

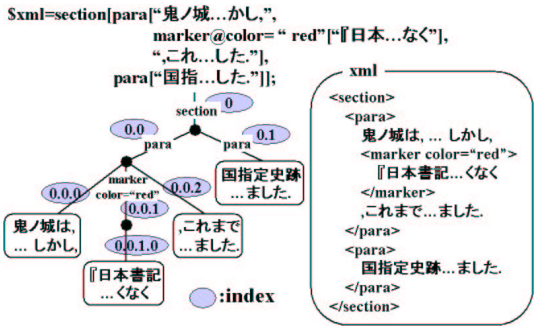

図 2: XML のデータモデル

2.2 インデックスとインデックススコープ

XML データモデルの基本となるシーケンスは，順序を持つデータ列であり各要素を指し示す index を持っている (図 3)．シーケンスの最初の要素の index は 0 である．また，index は負の数でもよく，このときは右から数える．つまり最後の要素が −1 である．

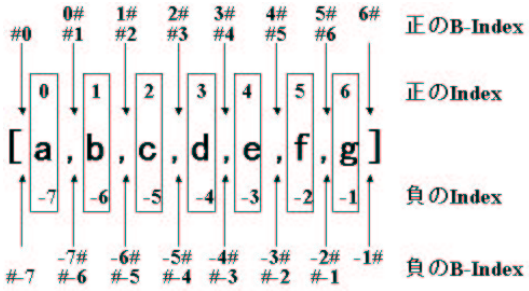

図 3: index の例

また xTrics においては，後述するシーケンスに対する様々な操作を行うことでインデックスは動的に変更される．これにより，意図したものと違う操作が行われることを防ぐため xTrics においてはインデックススコープを定義している．インデックススコープはインデックスが指し示す要素を常に指すものであり，インデックスの変更に関わらず同じ要素を指し示す．

インデックススコープは組み込み関数 scope() を用いて導出する．scope() は引数を 2 つ持ち，第 1 引数として対象となるシーケンスを指定する．第 2

引数としてはインデックスを指定する．例として，$str="今日は暑い．"; という文字列に対して，$iscope=scope($str,3); は「暑」に対するインデックススコープを得る．ここで,「今日は」の後に「とても」が挿入され$str の値が「今日はとても暑い.」に変わったとしても$iscope は「暑」を指し示す．しかし，もし$str の値が「今日は寒い.」に書き換えられた時は，$iscope は存在しないため破棄する．

2.3 インデクシングとスライス

シーケンス型の変数に対してインデックスおよびインデックススコープを指定して，指定の要素を抽出する演算をインデクシングと呼び，変数名 [index] で記述する．xTrics では変数を$value のように$のあとに任意の英数字の列で記述する．

表 1 はインデクシングの例である．ここで 4) は，index の範囲を超えるためエラーとなる．また 5) は，$seq[-2] の結果がシーケンスであるため，さらにインデクシングを行える．しかし 6) は，$seq[2] の結果がシーケンスではないためエラーとなっている．

$seq=[1,‘x’,2.4,“xTrics”,‘a’];	
1) $seq[2];	2.4
2) $seq[-2];	“xTrics”
3) $seq[0];	1
4) $seq[-10];	error
5) $seq[-2][-5];	‘T’
6) $seq[2][4];	error

表 1: インデクシングの例

また，シーケンス型の変数に対して範囲を指定することで，部分シーケンスを抽出する演算をスライスと呼び，変数名 [index1:index2] で記述する．

ここで index1 と index2 の関係は index1 ≤index2 となる必要があり，この時の戻り値は指定した index を含むシーケンスとなる．もし index1 >index2 のときはエラーとなる．ただし負の index の場合は正の index に変換して比較する．

指定する index には，インデックスとインデックススコープの他に要素と要素のあいだを指し示す B-index(Between index) も使用することもできる．

表 2 はスライスの例である．ここで 2) は，8>5 であるためエラーとなる．また 4) は，index の範囲を超えているためエラーとなる．

$str=“xTrics はエックストリックスと読む”;	
1) $str[0:5];	“xTrics”
2) $str[8:5];	error
3) $str[6#:-4];	“エックストリックス”
4) $str[20:25];	error
5) $str[6#:#7];	[ ]
6) $str[6:#7];	“は”

表 2: スライスの例

次に XML のインデクシングとスライスについて述べる．XML は，木構造であるので，親 index. 子 index. 孫 index.· · · のように ‘.’ で連結された path で index を記述する．また，XML はラベルつきシーケンスでもあるため，index の変わりにラベルを指定しても良い．このとき指定するラベルは，タグ名，属性名，属性名+属性値の任意の組み合わせでよく，ラベルとマッチする要素が選ばれる．もしマッチするラベルが複数あるときは，ラベル (index) で指定する．ここで，番号を省略すると最初にマッチした要素が選ばれる．例として，図 4 において，$XML[0.1] と $XML[label0.label1]，$XML[0.label1] はすべて同値となる．また，XML はシーケンスの入れ子構造でもあるため，$XML[0.1.1] は$XML[0][1][1] と同値となる．

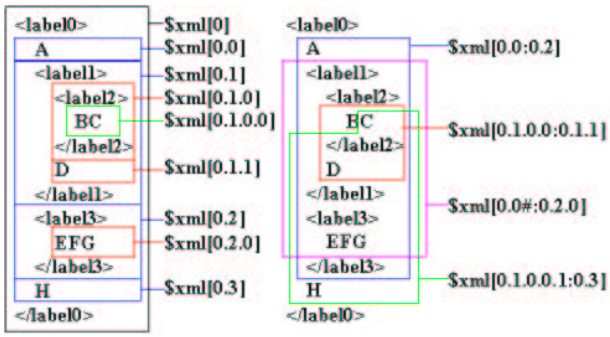

図 4: XML 型のインデクシングとスライス

XML のスライスにおいては，抽出された値が林となる場合がある．しかし xTrics における XML には明確な順序があるため，抽出された値は XML を要素とするシーケンスになる．またスライスで抽出された XML は，一見するとウェルフォームドになってないように見えるが，xTrics において XML は木構造で処理するため，抽出された XML は必ずウェ

ルフォームドとなる．

2.4 シーケンスパターンとマッチング操作

xTrics ではシーケンスに対して要素の出現パターンを記述し，そのパターンとのマッチング操作を行うことで任意のパターンを持つ要素すべてを置換したり，削除することが出来る．この節では，xTrics におけるシーケンスパターンとマッチング操作について述べる．

2.4.1 シーケンスパターン

シーケンスパターンは，スキーマ言語によく見られる「繰返し (*，+，?)」や「選択 (|)」などの正規表現の記法を各要素に適応して記述する (演算子は表 3 に参照されたい)．

*	0 回以上の繰り返しにマッチさせる．
+	1 回以上の繰り返しにマッチさせる．
?	0 または 1 回の繰り返しにマッチさせる．
{m,n}	m から n 回の繰り返しにマッチさせる．
!	任意の要素 1 つにマッチさせる．
(···)	() 内をグループ化する．
A|B	A または B にマッチするパターンを生成する．
\	パターン演算子 (*，+，?，{，}，|) 等の文字にマッチできるようにする．
<···>	<>内のパターンは自分を含む子孫要素とマッチする．

表 3: パターン演算子

例えば [a*,b,c] のようなシーケンスパターンがあったとき，このパターンは [b,c] ，[a,b,c] ，[a,a,b,c] ，··· にマッチする．

また，文字列についても同様に記述する．つまり，[’今’,’日’,!*] のように記述して,「今日」で始まる文字列にマッチする．しかし，このような記述では冗長であるので文字列パターンは"今日!*"と記述できる．ここで，文字列パターンにおいて例えば「1+2」とマッチさせたい場合，"1+2"と記述したのでは 12 や 112 とマッチしてしまう．そこで「*」や「+」のようなパターン演算子を文字列としてマッチさせるには「*」や「\+」のように記述する．つまり文字列パターン「1+2」は"1\+2"と記述する．

次に XML シーケンスにおけるシーケンスパターンについて述べる．xTrics において XML はラベル付きのシーケンスと記述するため，ラベルについてもパターンを指定できることが望ましい．ラベルにおいては，タグ名，属性名，属性値の各値に対して文字列パターンと同様の記述をする．またラベルパターンは,「タグ名」,「属性名」,「属性名+属性値」の任意の組み合わせでよい．つまり，(年月日 |@年月日)[!*] のように記述でき，この例ではタグ名か属性名が「年月日」であるような枝がマッチする．

ここで XML は，入れ子シーケンスで表現されるため，ある要素の子孫にパターンが出現するようなパターンを求めたい要求がある．そこで xTrics においては<>を使って記述する．つまり，論文 [<著者 ["!*岡本!*"]>] のように記述すると，論文タグの子孫に著者タグを持ちその要素値に「岡本」を持つような枝とマッチする．

2.4.2 マッチング操作

マッチング操作は組み込み関数 matching() を使用して行う．matching() は引数を 2 つ持ち，第 1 引数としてマッチング操作を行う対象となるシーケンスを指定する．第 2 引数としてはシーケンスパターンを指定する．戻り値としてマッチしたシーケンスの範囲のシーケンスを得る．ここでシーケンスの範囲とは，マッチしたシーケンスの最初と最後のインデックススコープとする．以下に図 2 の XML データに対して，マッチング操作を行うの例を示す．

1) $match=matching($xml,"鬼の城");
   $match => [0.0.0:0.0.2 , 0.1.8:0.1.10]

2) $match=matching($xml,para[<"!*築城!*">]);
   $match => [0.0]
   $xml[$match[0]] =>
   <para>
   鬼ノ城は，663 年の白村…
   ……
   …いわれてきました．
   </para>

ここでマッチング操作の結果に対しては，2) のようにインデクシングやスライスをすることでマッチ

したデータを得ることも出来る．

また xTrics には，シーケンスパターンを利用してシーケンスに対して，分割や置換を行うことも出来る．

分割操作は組み込み関数 split() を使用して行う．split() は引数を 2 つ持ち，第 1 引数として分割操作を行う対象となるシーケンスを指定する．第 2 引数としてはシーケンスパターンを指定する．戻り値として分割されたシーケンスのシーケンスを得る．分割操作はシーケンスに対して行うため，XML に対しても行えるが，このときはマッチしたデータを中心に右の木と左の木に分割され，先祖は両方の木が持つ形となる．一般に分割操作は文字列に対して適用する．以下に文字列と図 2 の XML でーたに対して分割操作を行うの例を示す．

1) $name="岡本 辰夫";
   $data=split($str," ");
   $data => ["岡本" , "辰夫"]

2) $data=split($xml,marker[!*]);
   $date =>
   [section[para["鬼ノ城は，…しかし，"]],
     section[para[", これまで…きました."],
       para["国指定史跡…きました."]]
   ]

置換操作は組み込み関数 sub() を使用して行う．sub() は引数を 3 つ持ち，第 1 引数として置換操作を行う対象となるシーケンスを指定する．第 2 引数としてはシーケンスパターンを指定する．第 2 引数としては置換するデータを指定する．戻り値として置換されたシーケンスを得る．以下に置換操作を行うの例を示す．

1) $str="今日は熱い．私は熱いのは苦手だ．";
   $data=sub($str,"熱い","暑い");
   $data =>"今日は暑い．私は暑いのは苦手だ．"

2.5 XML の操作

xTrics における XML の操作としては，前述したインデクシング，スライス，マッチング操作のほかに XML の木構造に対する挿入，削除の操作と各ノードの持つタグ名の変更や属性値の追加，変更，削除がある．

2.5.1 挿入操作

挿入操作は組み込み関数 insert() を使用して行う．insert() は引数を 3 つ持ち，第 1 引数として挿入操作を行う対象となる XML を指定する．第 2 引数としては B-index をとり，その場所にデータを挿入する．第 3 引数として挿入するデータを指定する．

また挿入操作は，XML だけでなくシーケンスに対しても挿入することができる．

2.5.2 削除操作

削除操作は組み込み関数 delete() を使用して行う．delete() は引数を 2 つ持ち，第 1 引数として削除操作を行う対象となる XML を指定する．第 2 引数としては index か範囲指定をとり，その場所のデータを削除する．範囲指定は 2.3 節のスライスと同様に start:end で表記する．ここで start と end は index か B-index をとる．

削除も挿入と同様にシーケンスに対して適用可能である．

2.5.3 タグ名や属性値の操作

XML の各ノードにおいてタグ名は必ず 1 つである．そのため，タグ名に関する操作は変更のみである．タグ名の変更は組み込み関数 ChangeElement() を使用する．ChangeElement() は引数を 3 つ持ち，第 1 引数として操作を行う対象となる XML を指定する．第 2 引数としては目標のノードを指定する index をとる．第 3 引数として変更するタグ名を文字列で指定する．

また，各ノードにおける属性は複数存在するので，属性に関する操作は追加，変更，削除がある．属性の追加，変更は組み込み関数 ChangeAtt() を使用し，属性の削除は組み込み関数 DeleteAtt() を使用する．

ChangeAtt() は引数を 4 つ持ち，第 1 引数と第 2 引数は ChangeElement() と同じデータをとる．第 3 引数として属性名を文字列で指定する．もし対象のノードに指定した属性名が存在しなければ，新しい属性を追加し，存在すれば属性値を変更する．第 4 引数としては，属性値を文字列で指定する．

DeleteAtt() は引数を 3 つ持ち，第 1 引数，第 2 引数，第 3 引数は ChangeAtt() と同じデータをとる．ここで第 3 引数は削除する属性名である．

2.6 制御フロー

xTrics は，ほかの言語で知られている普通の制御フロー文を備えている．この節では，その中でもっとも一般的な分岐と繰返しについて述べる．

2.6.1 分岐

xTrics における分岐の制御フローは if 文のみを備えている．if 文は以下のような構文で定義されている．

$$
\begin{aligned}
\textit{if}\ \text{文} \quad &:= \quad \textit{if}\ \text{条件式}\ (\textit{then} \mid :)\ \text{式} \\
&\quad\ \ [\textit{elif}\ \text{条件式}\ (\textit{then} \mid :)\ \text{式}]* \\
&\quad\ \ [\textit{else}\ \text{式}] \\
\text{式} \quad &:= \quad \text{文;} \mid \textit{begin}\ (\text{式})*\ \textit{end;} \mid \{\ (\text{式})*\ \}
\end{aligned}
$$

このように if 文には 0 個以上の elif 部があってもよく，else 部を付けてもよい．if···elif···elif··· 列は，ほかの言語で見られる switch 文や case 文の代用品として使える．

2.6.2 繰返し

xTrics における繰返しの制御フローには while 文と for 文を備えている．また，繰返し内のみ有効な制御フローとして break 文，next 文，redo 文がある．これらの制御フローは以下のような構文で定義されている．

$$
\begin{aligned}
\textit{while}\ \text{文} \quad &:= \quad \textit{while}\ \text{条件式}\ (\textit{do} \mid :)\ \text{式} \\
\textit{for}\ \text{文} \quad &:= \quad \textit{for}\ \text{変数}\ \textit{in}\ \text{シーケンス}\ (\textit{do} \mid :)\ \text{式} \\
\textit{break}\ \text{文} \quad &:= \quad \textit{break;} \\
\textit{next}\ \text{文} \quad &:= \quad \textit{next;} \\
\textit{redo}\ \text{文} \quad &:= \quad \textit{redo;}
\end{aligned}
$$

ここで，while 文は条件式が真のあいだ式を実行していき，for 文はシーケンス内の各要素に対して，それらがシーケンスに現れる順序で繰返しをする．break 文はもっとも内側のループを脱出する．next 文はもっとも内側のループの次の繰り返しにジャンプする．redo 文はループ条件のチェックを行なわずに，現在の繰り返しをやり直す．

2.7 操作の論理単位

xTrics は個人化に関する研究のなかで開発したもので，複数の利用者の操作を組み合わせて動作させることも考えられるため，それぞれの操作は独立に動作できるように考案している．

そのため，xTrics におけるすべての文は，処理後に成否が判定される．ここで成功していれば，その文で行った処理の結果をデータに反映し，もし失敗していれば，その文で変更されたデータはすべて破棄されて，文実行前のデータに戻される．

また xTrics では成否にかかわらずプログラム終了まですべての文を処理する．つまり，途中で失敗した文が現れてもその文を無効にするだけで，その後の文も順番に処理する．例えば，図 2 の XML データに対して

```
1) $xml=insert($xml,#0.para,title["鬼ノ城"]);
2) $xml=delete($xml,0.para(-1));
```

のような xTrics があったとき，1) は"鬼ノ城"というタイトルを入れる操作であり，2) は最後のパラグラフを消す操作である．この 2 つに明確な関係はなくお互いの操作は独立しているといえる．

しかし，例えばの次のようにある範囲にタグを追加する xTrics を考える．

```
1) $str=$xml[0.para(-1).0.28:0.para(-1).0.61];
2) $xml=delete($xml,
        0.para(-1).0.28:0.para(-1).0.61);
3) $xml=insert($xml,#0.para(-1).0.28,
        marker@color="blue"[$str]);
```

このとき，もし 1)，2) が成功して 3) が失敗すると指定範囲の文字列が消えてしまう．このため明確な関係のある操作を 1 つの論理単位として，すべて成功するか試行する処理が必要となる．このため xTrics においては，試行をする try 文を以下のように定義している．また，try 文内で明示的に例外を投げる exception 文も定義している．

$$
\begin{aligned}
\textit{try}\ \text{文} \quad &:= \quad \textit{try}\ \text{式} \\
\textit{exception}\ \text{文} \quad &:= \quad \textit{exception;}
\end{aligned}
$$

ここで try 文は，式内のいずれかの文が失敗するか明示的に例外が投げられると，この試行は失敗として，試行前の状態に戻す．つまり前例においてtry{1)2)3)} としておけば，もし 3) で失敗しても文字列が消えてしまうことがなく，すべて成功すれば範囲にタグを追加することができる xTrics となる．

3　既存の操作言語との比較

xTrics は XML 操作スクリプト言語であり，XML 文書を他の XML 文書に変換する機能を持っている．同様の機能を持つものとして XSLT がある．

XSLT は，XML 用のスタイルシート言語 XSL から XML 文書の構造の変換機能だけを取り出したものであり，強力な変換機能を持つ．しかし，XSLT には次のような問題点がある．

1) 変換する文書の構造が複雑になればなるほど，その記述は加速度的に煩雑になる．
2) 文字列内に含まれる特定の単語やパターンを簡単に置換する機能はない．

最初の問題は，XSLT では変換する文書の構造を宣言的にテンプレートという形で記述するが，変換後の文書構造が入り組み，可読性が低下し記述も煩雑になる．xTrics では，操作を手続的に列挙する方式なので変換する構造が大きき変わると処理の記述は多くなるが，手続的に処理を記述するため見直しやデバッグが容易である．また xTrics においては，関数として手続きをまとめることで，似たような操作を何度も使う時の記述は簡単である．

次に，文字列内に含まれる特定の単語やパターンを簡単に置換する機能については，XSLT ではパターンの抽出に XPath[3] のサブセットを使用しているが，XPath の基本仕様には，特定の単語を得る記述はないため，XSLT では特定の単語やパターンを簡単に置換する機能はない．xTrics では，文字列も XML もシーケンスとして簡単にパターンマッチングが出来るので，特定の単語やパターンを置換することも容易にできる．

4　おわりに

本研究では，簡単な記述で XML 文書を操作することができ，文字列などに対しても強力な操作をもつ XML 操作スクリプト言語 xTrics を提案した．

しかし，現状ではプログラミング言語としての xTrics と XML 文書の記述に大きな隔たりがあるので，今後は xTrics の XML 表現である xTrix の開発に力を入れたい．

また，これまで xTrics による個人化やオントロジーに関する研究を行ってきたが，今後更にこれらの研究と，それに有効活用できる xTrics の検討を行いたい．

そして，xTrics においてはパターンマッチングを XML の木構造だけでなく文字列にも拡張しているが，この場合例えば,「岡山県立大学」と「岡本」の両方の文字列を含むパターンを記述したいという要求も出てくる．しかし現在のパターンでは複雑になるため，パターン記述に関してもっと検討していきたい．

参考文献

[1] 藤野猛士, 石崎勝俊, 谷本奈緒美, 細田昌明, 國島丈生, 横田一正, “ユーザ適応型マルチメディア情報提示システムの実現”, 電子情報通信学会データ工学ワークショップ, B4-7, 倉敷,2002 年 3 月 4-6 日.

[2] 岡本辰夫, 吉田奈美子, 國島丈生, 横田一正, “XML 操作スクリプト言語による個人化手法の提案”, 電子情報通信学会データ工学ワークショップ, A6-4, 倉敷,2002 年 3 月 4-6 日.

[3] XSL Transformations (XSLT) Version 1.0, http://www.w3.org/TR/xslt,1999 年 11 月 16 日.

[4] XML Path Language (XPath) Version 1.0, http://www.w3.org/TR/xpath,1999 年 11 月 16 日.