構造化文書の依存関係記述のための拡張スキーマ

An Extended Schema for Describing Dependencies in Structured Documents

林康史，劉東喜，中野圭介，胡振江，武市正人
Yasushi HAYASHI, Dongxi LIU, Keisuke NAKANO, Zhenjiang HU, Masato TAKEICHI

東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, University of Tokyo
{hayashi,liu,ksk,hu,takeichi}@mist.i.u-tokyo.ac.jp

W3C XML Schema は，XML 文書の構造や形式の制約を記述するための代表的なスキーマ言語であるが，計算や変換を伴う要素間の依存関係の制約を記述することが難しい．本論文では，要素間の依存関係を与える計算や変換の記述を XML Schema の中に埋め込むことによりスキーマを拡張する．また，XML 文書内のデータに対話的な変更が加えられるような環境においても，データの変更に応じて計算や変換を動的に行うことで要素間の依存関係を保持するような計算機構を提案する．本機構では，双方向変換の枠組みを利用することにより，変換によって生成されたデータ側が変更された場合でも関係するデータが自動的に更新される．

1 はじめに

W3C XML Schema (WXS) [13] は，構造化文書である XML 文書の構造や形式の制約を記述するための代表的なスキーマ言語として広く使われており，XML の構造や形式を記述するために適している．しかし，WXS ではインスタンスである XML データ内の要素間についての計算や変換を伴う依存関係を記述することは難しい．例えば，「要素 C の値は，要素 A と要素 B の値の合計でなければならない」といった簡単な制約をかけることはできない．実際の XML 文書ではこのような依存関係が必要になる場合も少なくない．これまで Schematron [6] のようなルールベースのスキーマ言語により要素間の依存関係を与える方法がよく用いられてきたが，動的な更新を想定していないために XML 文書編集しながら依存関係の制約を保つようなエディタを実装することは難しい．

本研究では，動的な編集にも対応した依存関係を記述するための WXS の拡張方法を提案する．依存関係は XPath 式 [2] を用いて記述され，スキーマ内のある型を持つ要素が他の要素からどう計算されるかを自由に記述することができる．例えば，単価・注文数・小計・合計金額の情報を含む XML による注文書を考える．提案する手法では，小計や合計金額に相当する型の記述の中に「`(../unit)'*'(../number)`」（単価に注文数を掛ける）や「`sum(./subtotal)`」（小計を足し合わせる）のような XPath 式を記述することができる．この注文書を編集する場合，商品の注文数の変更に応じて小計・合計金額が更新されることが期待されるが，提案する手法では，データ内の編集された部分に影響される XPath 式を解析し，再計算を行うことで制約を維持することが実現される．

さらに，本稿で提案する手法では，双方向変換の枠組みに基づいた更新にも対応している．注文書のような依存関係の例では，小計や合計のような計算結果側を変更することは考えにくいが，実際の場面では，計算結果側の変更によって計算に使われた部分が更新されるような依存関係も存在する．例えば，文書の本文にあるセクション名を抽出して目次を付加した文書は，スキーマ内の目次部分の型としてセクションの抽出・変換処理を表現する XPath 式を用いて依存関係が記述される．この際，変換結果である目次側の編集を本文側のセクション名に反映させることができれば，より柔軟な XML 文書編集が可能になり，編集の効率と信頼性を高めることにつながる．本稿では，このような計算結果側の編集を許すような依存関係を**双方向型**と呼び，注文書の例のように双方向の更新を期待しない依存関係を**単方向型**と呼ぶ．我々の提案する拡張スキーマでは，単方向型と双方向型の両方を明示的に指定することでより信頼性の高い文書の設計を可能にしている．

本稿では，双方向変換言語 Bi-X [8, 10] とジャストシステムの開発した XML エディタ xfy を利用して実装されたシステムについても紹介する．本シス

テムでは，提案する拡張スキーマに記述された依存関係を崩すことなく編集・更新を自由に行うことができる．

本稿の構成は次の通りである．まず第 2 節において拡張スキーマを定義する．次の第 3 節では，双方向言語 Bi-X について概略を述べる．次の第 4 節で依存関係の保持機構について説明し，第 5 節で応用例を紹介する．第 6 節で関連研究について触れ，最後に第 7 節を本稿のまとめとする．

2 拡張スキーマの記述

WXS で記述できない依存関係を記述するために annotation 要素内の appinfo 要素を用いて WXS を拡張する．appinfo 要素は WXS の中でコメントを記述するためのものであり，通常その内容は妥当性検証のシステムに無視されるが，Schematron [6] のようにその情報を読み取って何らかの処理に利用することも可能である．依存関係は，双方向の更新を伴わない単方向型の場合と双方向の更新を伴う双方向型の場合を区別して記述する．数値の計算等の単方向型の依存関係を記述するためには，psd:calc 要素に，インスタンス文書内の要素を指定する二つの XPath 式 *xpath1*，*xpath2* を用いて次のように記述する．

```
<psd:calc expression=" funname( xpath1 )"
          target=" xpath2 "/>
```

xpath1 で参照先要素の集合を指定し，*funname* で参照先の要素を引数とする関数を記述する．target 属性の値である *xpath2* は，評価式 *funname*(*xpath1*) の評価結果の出力先を指定している．つまり関数 *funname* が，*xpath1* で指定される参照先要素の値と *xpath2* で指定される出力先要素の値との依存関係を規定していると考え，参照先の値が変更されれば再計算によって出力先要素の値が更新されることを要求している．この psd:calc 要素は，target 属性で指定される出力先要素が宣言されている WXS 内の場所に埋め込まれる．例えば，成績表のデータを持つ XML 文書のスキーマ内で受験者の各得点と合計を含む次のような student 要素が宣言されている場合に，各科目の合計を total 要素に出力するには，次のように記述する．

```
<xsd:elment name="student">
 <xsd:complexType name="studentType">
  <xsd:sequence>
   <xsd:element name="name"
                type="xsd:string"/>
   <xsd:element name="japanese"
                type="xsd:string"/>
   <xsd:element name="mathematics"
                type="xsd:string"/>
   <xsd:element name="english"
                type="xsd:string"/>
   <xsd:element name="total"
                type="xsd:string">
   <xsd:annotation>
    <xsd:appinfo>
     <psd:calc expresion="../japanese +
                          ../mathematics +
                          ../english"
       target="/students/student/total"/>
    </xsd:appinfo>
   </xsd:annotation>
  </element>
 </complexType>
<xsd:element>
```

これにより，XML インスタンス文書内の次のような student 要素の total 要素の中に合計点の値が生成される．

```
<student>
  <name>Hayashi</name>
  <japanese>85</japanese>
  <mathematics>70</mathematics>
  <english>90</english>
  <total>245</total>
</student>
```

この後，もしある得点の値が変更された時には再計算されて合計点の値が更新される．

双方向の更新を伴う双方向型の依存関係を記述するためには，psd:calc 要素の代わりに psd:bitrans 要素を用いて，

```
<psd:bitrans expression=" funname( xpath1)"
             target=" xpath2"/>
```

のように記述する．*xpath1* が参照先要素の集合を指定し，その集合を引数とする関数 *funname* は，後述の言語 Bi-X により記述されているものとする．参照先は要素を子供に持つ要素であってもよい．target 属性

の値である *xpath2* は，評価式 *funname*(*xpath1*) の評価結果の出力先を指定している．つまり関数 *funname* が，*xpath1* で指定される参照先要素と *xpath2* で指定される出力先要素との依存関係を規定していると考え，参照先の要素の内容が変更されれば再計算によって出力先要素の値が更新されることを要求している．また出力先要素の内容が変更されても逆計算が実行されて参照先要素の内容が更新されることも要求している．psd:calc 要素と同様に，psd:bitrans 要素は，target 属性で指定される出力先要素が宣言されている WXS 内の場所に埋め込まれる．例えば，全受験者の成績データを持っている XML 文書に，上位何人かを抽出した成績上位者のデータを追加した文書を考える．成績上位者のデータは全受験者を点数でソートして上位何人かを抜き出すという双方向変換によって作成することができる．これは，全受験者リストの部分と成績上位者リストの部分の間に整合性を要求する双方向型の依存関係を記述していると考えることができる．例えば，成績上位者リストを XML インスタンス文書の toprank 要素に出力するには，拡張スキーマ内において，全受験者のデータを持つ scores 要素の型が宣言されている部分に次のように記述する．

```
<xsd:element name="scores">
 <xsd:complexType name="scoresType">
  <xsd:sequence>
   <xsd:element name="students"
                type="studentsType"/>
   <xsd:element name="toprank"
                type="toprankType">
    <xsd:annotation>
     <xsd:appinfo>
      <psd:bitrans
           expresion="f_top(../students)"
           target="/scores/toprank" />
     </xsd:appinfo>
    </xsd:annotation>
   </xsd:element>
  </xsd:sequence>
 </xsd:complexType>
</xsd:element>
```

これは全受験者の得点データを含む students 要素を参照して，それに成績上位者リストを作る関数 f_top を適用し，結果を toprank 要素に出力することを記述している．これにより，

```
<scores>
  <students>
    <student>
      <name>Hayashi</name>
      <japanese>85</japanese>
       ...
    </student>
    ...
    <student>
      <name>Liu</name>
      <japanese>95</japanese>
        ...
    </student>
  </students>
  <toprank></toprank>
<scores>
```

のような全受験者の成績データを含む XML インスタンス文書内の toprank 要素に次のように成績上位者のリストが生成される．

```
  <toprank>
    <student>
      <name>Liu</name>
      <japanese>95</japanese>
        ...
    </student>
    <student>
      <name>Hayashi</name>
      <japanese>85</japanese>
        ...
    </student>
  </toprank>
```

この後，students 要素内でデータの変更があった場合には，toprank 要素内が変更されたデータに基づく成績上位者リストになり，逆に，toprank 要素内でデータの変更があった場合には，まず students 要素内の相当するデータが更新され，それに基づく成績上位者リストが再生成されて toprank 要素内のデータが新しい成績上位者のものに置き換えられる．

psd:calc 要素と psd:bitrans 要素は何個でも埋め込むことができる．後で説明するように，参照関係の解析によって計算順序が決められるが参照関係が循環するかどうかは計算時にチェックされ，循環す

る場合にはエラーになる．また psd:bitrans 要素に関しては，psd:bitrans 要素の参照先の要素の内部に別の psd:bitrans 要素を埋め込んではいけない．

3 双方向言語 Bi-X

本節では，次節で説明する依存関係保持のための計算機構が利用している双方向言語 Bi-X についての概略を述べる．

3.1 双方向変換

ある XML データ (ソース) から，その中の必要な情報を抽出・整形した XML データ (ビュー) を変換によって生成する場合を考える．両方の XML データに含まれるデータを整合性を保ちつつ更新したい場合，通常は，ソースの中の更新したいデータを探し出して更新し，ビューへの変換をやり直すことでビュー内の相当するデータも更新することができる．しかし，情報を抽出・整形したビューの方を先に更新し，それをソースの方に自動的に反映させることができればその方が簡単な場合が多い．ソースからビューへの変換を**順変換**と呼び，それぞれの順変換ごとにその逆方向の変換 (**逆変換**) の定義を与えれば，このようなことが可能になるが，両方向の変換を整合性を維持しつつ保守することは難しい．特に，ここでいう逆変換は，一般の逆関数と異なりソースデータの情報を落とすような単射でない順変換を扱うため，逆方向の計算には更新された結果だけでなく元のソースデータに関する情報も必要となり，この計算の記述は容易でない．そこで我々は順変換プログラムを与えるだけで，更新されたビューと元のソースから更新されたソースを得るための逆変換プログラムが自動的に得られる言語 Bi-X を提案した [8, 10]．3.2 節で説明するような Bi-X プログラムを用いて生成されたビュー文書に対して，文字データの変更，要素の追加・削除等の編集を行った後，自動生成された逆変換プログラムを実行することにより，ソース文書の相当する部分に変更を自動的に反映させることができる．

3.2 Bi-X プログラム

Bi-X プログラムはそれ自体が XML になる．その構文の一部を次に示す．

X::= <xseq> *X1 X2 ... Xn* </xseq>
| <xid/>
| <xchild/>
| <xconst> *C* </xconst>
| <xmap> *X* <xmap>
| <xif> *X1 X2 X3* </xif>
| <xwithtag> *String* </xwithtag>
...

一般に，ある変換を記述する Bi-X プログラム *X* は，複数の変換 *X1* ,*X2* ,...,*Xn* を合成することで定義され，<xseq>*X1 X2 ... Xn* </xseq>と記述される．このプログラムは，入力にまず変換 *X1* が適用され，その結果に対して変換 *X2* が適用されるというように順に変換が適用され，最後の変換 *Xn* の結果を *X* の出力とすることを意味する．また，構成要素になる基本的な変換も定義されており，例えば<xid/>は入力と同じものを出力する恒等変換，<xchild/>は要素の中身を出力する変換，<xconst>*C* </xconst>は入力に関係なく *C* を出力する変換，<xmap>*X* </xmap>は変換 *X* をそれぞれの要素に適用する変換である．簡単な Bi-X プログラムの例として，入力要素の子供要素から title 要素だけを抽出するものを次に示す．

```
<xseq>
  <xchild/>
  <xmap>
    <xif>
      <xwithtag>title</xwithtag>
      <xid/>
      <xconst/>
    </xif>
  </xmap>
</xseq>
```

これは，<xchild/>でまず入力要素の中身を列として取り出し，<xmap>でそれぞれの要素に対して，条件式のための<xif>を用いてタグ名が title かどうかチェックをし，title 要素であれば，<xid/> でそのまま残し，title 要素でなければ，<xconst/>で取り除いている．

4 依存関係保持のための計算機構

拡張スキーマによって加えられる依存関係の制約には，psd:calc 要素の記述と psd:bitrans の記述の二種類あるが，それらの依存関係保持のための計算機能は別々に実現される．psd:calc 要素の記述の方は，例えば，XForms の計算エンジンについて解

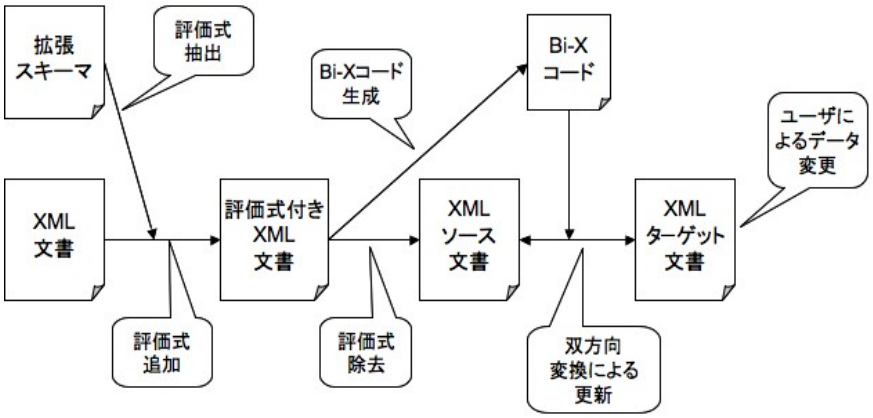

図 1: 処理の流れ

説した [1] 等にしたがえば実装することができ，また以下に説明する双方向型の psd:bitrans の記述の方の特別な場合とも考えられるので，ここでは双方向型の場合のみを説明する．

4.1 計算機構の概略

双方向型の依存関係を保持する仕組みは，3 節で説明した双方向変換に基づいている．処理の流れを図示したものが図 1 である．始めに，拡張スキーマと，拡張されている部分を無視した WXS を満たす XML 文書が与えられているとする．すなわち，埋め込まれた評価式の結果の出力先となっている XML 文書内の要素は空になっていてもよい．

本計算機構では，まず，拡張スキーマから埋め込まれた appinfo 要素内の情報を抽出し，bitrans 要素の target 属性で指定された出力先に, bitrans 要素の expression 属性の値である評価式を属性として追加する．その後それぞれの評価式は 4.2 節と 4.3 節で説明する方法で Bi-X プログラムに変換される．これが終わると XML 文書から，追加された属性を削除して元の文書に戻し，双方向変換のソース文書とする．このソース文書に生成された Bi-X プログラムを適用することによりビュー文書が得られ，これが拡張スキーマの依存関係を満たした文書となっている．4.4 節で説明するように，この文書に対して変更が加えられた場合，変更された部分が，埋め込まれた評価式の参照先のデータであっても評価結果の出力先のデータであっても，まず逆変換によってソース文書に変更が反映される．その後，そのソース文書を元に順変換を適用し，依存関係が保持された新しいビュー文書が生成される．

4.2 XPath の Bi-X コードへの変換

bitrans 要素の expression 属性値である評価式は $funname(xpath)$ の形をしているが，関数 $funname$ は, 関数定義ファイルの中で，Bi-X で定義するためすでに双方向変換プログラムになっており，評価式全体を双方向変換プログラムにするためには，XPath 式の部分をソース文書から参照先要素を得る Bi-X コードに変換すればよい．XPath の各ステップ $/nm_i[pred_i]$ は Bi-X で

```
<xmap>
  <xseq>
    <xchild/>
    <xif>
      <xwithtag>nm_i</xwithtag>
      <xid/>
      <xconst/>
    </xif>
    <xif>pred_i
      <xid/>
      <xconst/>
    </xif>
  <xseq>
</xmap>
```

と書ける．これを簡単に，

```
<xpathstep>
  <name>nm_i</name>pred_i
```

```
</xpathstep>
```

と表すと，XPath 式 $/nm_1[pred_1]/\ldots/nm_n[pred_n]$ は，

```
<xpathstep>
  <name>nm1</name>pred1
</xpathstep>
...
<xpathstep>
  <name>nmn</name>predn
</xpathstep>
```

と Bi-X プログラムにエンコードできる．

4.3 コードの合成

funname を *xpath* に適用するには，4.2 節の方法で得られた XPath 式部分の Bi-X コードに続いて次のように書く．

```
<xstore>$xpath</xstore>
<xfuncapp>
  <name> funname </name>
  <args>
    <xload>$xpath</xload>
  </args>
</xfunapp>
```

これは，XPath 評価の結果を，xstore 要素で変数 $xpath に入れ，xfunapp 要素の中の xload 要素で取り出して関数 *funname* を適用している．例えば，成績表の例の評価式を f_top(/scores/students) とすると/scores/students を 3.2 節にしたがって Bi-X にエンコードした後に上の関数適用の記述で *funname* を f_top に変えたものを続けて書けばよい．

この結果を target 属性で指定された要素の内容として出力するコードを続けて書く．ただし target 属性で指定された，ソース文書内の出力先要素に id 属性で固有の属性値ををつけておく．例えば，target 属性値が/scores/toprank である場合は，ソース文書内のその toprank 要素に id 属性をつけその値を 1 としておき，次のように書くと結果が toprank 要素内に生成される．

```
<xstore>$result</xstore>
<xload>$src</xload>
<xputid>
  <pos>1</pos>
  <xload>$result</xload>
</xputid>
```

これは評価式の結果を xstore 要素で変数$result に入れ，xputid 要素の中で取り出している．xputid は，<xload>$src</xload>で取り出されたソース文書中で pos 要素の値を属性値に持つ要素に，二番目の引数の値 (ここでは$result) を出力する．このようにして，一つの評価式を実行して結果を更新するまでの Bi-X コードが得られる．

評価式が複数ある場合，評価式間に参照の依存関係が存在しうるのでそれを考慮して正しい順序で実行する必要がある．ここで正しい順序とは，どの評価式に対してもそれが再評価される前に，その評価式内で参照している先がすでに再評価されているような順序とする．このような計算順序を決めるために，参照関係を表す有向グラフである依存グラフを作成してトポロジカルソートのアルゴリズムを利用して順序を決定すればよい (例えば，[1] を参照)．評価式が n 個あって, それらに対応する Bi-X コードを正しい順序に並べたものを $X_1, \cdots, X_n$ とすると，これらを単に<xseq>要素を用いて並べて合成することより，ソース XML 文書を受け取って，正しい順序ですべての評価式を実行する一つの Bi-X プログラムが

$$\texttt{<xseq>}X_1 \cdots X_n\texttt{</xseq>}$$

のように得られる．

4.4 双方向更新

評価式が一つの場合は，それから生成された Bi-X コードを X とすると，ソース文書を受け取って X を実行することにより，計算結果が反映されたビュー文書が出来上がる．これが拡張スキーマの対象となる XML 文書であり，これに対してデータの変更が加えられる．ビュー文書が変更された後，依存関係をみたすように再評価するには，変更された文書に対して逆変換 X^{-1} を実行し，得られた新しいソース文書に対してさらに順変換 X を行う．変換は XPath の評価部分 ($getpath$), 関数適用部分 ($funapply$)，結果出力部分 ($putid$) の三つの部分に分けられ，その逆変換は，($putid^{-1}$)，($funapply^{-1}$)，($getpath^{-1}$) の順で実行される．変更箇所が X の参照先に相当する部分である時は，XPath 評価の逆変換のところ $getpath^{-1}$ でソース文書に変更が反映され引き続いておこる順変換により，新しいソース文書に対して再評価が行わ

れ，新しいビュー文書に反映される．成績表の例では，ターゲット文書内の students 要素内のデータが変更された場合には，まずそれが逆変換でソース文書内の students 要素内のデータに反映され，そのデータをもとに students 要素内のデータと toprank 要素が作り直されて新しいビュー文書になる．また toprank 要素内のデータが変更された場合には $funapply^{-1}$ と $getpath^{-1}$ でソース文書内の students 要素内のデータに反映され，後は students 要素内のデータに変更された時と同様に students 要素内のデータと toprank 要素が作り直されて新しいビュー文書になる．

この仕組みは評価式が複数あっても同様に働く．Bi-X プログラム X はソース文書を受け取って，$X_1, X_2, \ldots, X_n$ の順に個々の変換の順変換を行いながら，それぞれの結果をソース文書内の出力先に出力していき，すべての評価結果が反映されたビュー文書が出来上がる．ビュー文書が編集された後，依存関係をみたすように再評価するには，まずこの逆変換 X^{-1} を実行し，得られた新しいソース文書に対してさらに順変換を行う．逆変換 X^{-1} は更新されたビュー文書を受け取って $X_n^{-1}, X_{n-1}^{-1}, \ldots, X_1^{-1}$ の順で個々の変換の逆変換を行いながら変更をソース文書に反映させていく．引き続いておこる順変換により，新しいソース文書に対して依存関係を満たすように個々の評価式の再評価が行われ，最後のビュー文書に反映される．

5 実装

4 節で説明した依存関係保持のための計算機構をジャストシステムの XML アプリケーション開発環境である xfy [7] を使って実現した．xfy は，XSLT と似た独自の変換言語 XVCD と専用のブラウザを用いることによって，様々な XML ボキャブラリをブラウザ上で表示し，それを編集する GUI を定義することにより，容易に自分自身の XML アプリケーションを構築することができる．また拡張性に優れ，新しい機能を Java で実装して xfy のプラグインとして組み込むことが可能であるので，4 節で説明した処理を自動的に実行するコマンドをプラグインとして組み込むことで，ユーザが行うブラウザ上の操作から起動できるようにした．

ここではこれまで説明で用いた試験成績の例を xfy 上で実際に作成したものについて述べる．試験成績データを元にして，これまで説明したように拡張ス

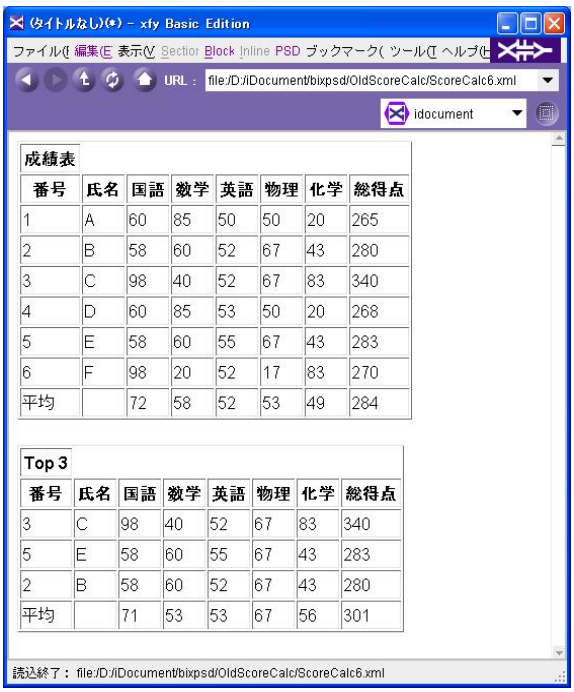

図 2: 成績表

キーマを作成し，ブラウザ上の編集操作に対して，記述された依存関係を保持するように XML データを自動的に更新する機能を実現する．受験者の氏名・番号，各科目の点数が記録された XML データを元に，XVCD による表示定義に従って xfy のブラウザで表示したものが図 2 である．受験者の合計点と各科目ごとの平均点は計算によって生成されるが，ブラウザ上で科目ごとの得点が変更されたときには，再計算によってデータを更新し，それがブラウザ上で表示される．また，受験者の総得点による成績上位 3 人のリストは 2 節で説明したような psd:bitrans 要素の記述に基づいて生成されているため，全受験者リストの部分と成績上位者リストの部分の，どちらのデータを変更しても，他方の相当するデータが同期して変更され，総得点と平均の値も計算し直される．さらに成績上位者リストも更新された値によって作り直され，その平均点が計算される．

6 関連研究

WXS の記述力の限界を認識し，他のスキーマ言語，特に Schematron と組み合わせて使う方法はいろいろと試みられている [3, 12]．WXS を拡張する研究としては，[11, 14] 等があり，[11] の SchemaPath では WXS を，XPath パターンに基づく条件式が記述で

きるように拡張し，条件式を用いた要素間の制約を記述できるようにしている．[14] では, 制約のルールの記述に XQuery を使えるように拡張することを提案している．これらの目的は WXS でチェックできない制約をチェックすることであり，制約を満たすようにデータを生成したり，データの変更に応じて他のデータを更新したりすることは考えていない．

XPath で記述された計算や制約に基づいて XML データを生成したり更新したりするシステムは既存の XForms[4] の実装や xfy[7] 等で実現されており，[1] はその計算機構について説明している．これらの計算機構は XPath が参照するものは数値や文字データに限られており，要素を子供に持つような構造を持つ要素を参照することはできない．

我々は，[9] で，要素を子供に持つような構造を持つ要素を参照するコードを XML 文書に埋め込むことにより，計算による依存関係を保持する環境を提案しているが，そこでは双方向の更新については考えていない．また，[5] では，XML データベースのソース文書から，Bi-X を用いてある種の依存関係を保持することが可能なビューを生成しているが，この場合は，本稿で述べた機構のように要素間の依存関係を，XPath 参照を用いて個別に記述することはできない．

7 まとめと今後の課題

本論文では， WXS の中に，依存関係を定義する計算や変換の記述を埋め込むことによりスキーマを拡張した．また XML 文書内のデータに対話的に変更が加えられるような状況において，データの変更に応じて動的に計算や変換を行いスキーマで定義された依存関係を保持する機構を提案した．特に双方向の更新が可能な依存関係を保持する機能を双方向変換言語 Bi-X を用いることによって可能にした．さらに，xfy を用いてこのような保持機能を実装した．

今後の課題は，拡張スキーマを生成・編集をサポートする環境を提供することである．現在は，依存関係の記述の中で `target` 属性で出力先場所を XPath で指定する必要があるが，スキーマ内の依存関係記述が埋め込まれた位置から自動的に生成できれば，`target` 属性の記述は不要になる．また，参照先の XPath の記述もスキーマ構造のグラフィカルな表示の操作から自動的に生成されることが望まれる．そのような機能を備えたツールを開発することにより，より使いやすい環境を提供することが期待される．

謝辞

本研究で用いた xfy システムについての技術的な支援をしていただいたジャストシステム株式会社に感謝する．

本研究は文部科学省リーディングプロジェクトである「e-Society 基盤ソフトウェアの総合開発」の一環である「高信頼構造化文書変換技術」の中で行われた．

参考文献

[1] J. Boyer, M. Honkala, The XForms Computation Engine: Rationale, Theory and Implementation Experience. In *Proc. of the 6th IASTED International Conference, Internet and Multimedia Systems, and Applications, (IMSA 2002)*, 2002.

[2] J. Clark and S. DeRose (eds.), XML Path Language (XPath) Version 1.0. W3C Recommendation, November, 1999, `http://www.w3.org/TR/xpath`.

[3] R. L. Costello, XML Schemas: Best Practices. 2003. `http://www.xfront.com/BestPracticesHomepage.html`.

[4] M. Dubinko, J. Dietl, L. Klotz, R. Merrick and T.V. Raman (eds.), XForms 1.0. W3C Recommendation, 2006, `http://www.w3.org/TR/xforms/`.

[5] Y. Hayashi, D. Liu, K. Emoto, K. Matsuda, Z. Hu and M. Takeichi, A Web Service Architecture for Bidirectional XML Updating. In *Proc. of Joint Conference of the 9th Asia-Pacific Web Conference and 8th International Conference on Web-Age Information Management (APWeb/WAIM 2007)*, 2007, LNCS4505, Springer, pp.721–732.

[6] R. Jelliffe and Academia Sinica Computing Centre, The Schematron Assertion Language 1.5. `http://ascc.net/xml/resource/schematron/Schematron2000.html`.

[7] Justsystem Corporation, xfy Technology. `http://www.xfytec.com`.

[8] D. Liu, Z. Hu, M. Takeichi, K. Kakehi and H. Wang, A Java Library for Bidirectional XML Transformation. *JSSST Computer Software*, 24(2), 2007, pp.167–177.

[9] D. Liu, Z. Hu and M. Takeichi, An Environment for Maintaining Dependency in XML Documents. In*Proc. of ACM Symposium on Document Engineering (DocEng 2005)*, pp.42-51.

[10] D. Liu, Z. Hu and M. Takeichi, Bidirectional Interpretation of XQuery. In*Proc. of ACM SIGPLAN Workshop on Partial Evaluation and Program Manipulation (PEPM 2007)*, 2007, pp.21-30.

[11] P. Marinelli, C. Sacerdoti Coen, F. Vitali and SchemaPath, A Minimal Extension to XML Schema for Conditional Constraints. In *Proc. of the 13th International World Wide Web Conference*, 2004, pp.164–174.

[12] E. Robertsson, Combining Schematron with other XML Schema Languages. 2002, http://www.topologi.com/public/Schtrn_XSD/Paper.html.

[13] H. Thompson et al. (eds.), XML Schema part 1: Structures Second Edition. W3C Recommendation, October, 2004, http://www.w3.org/TR/xmlschema-1/

[14] P. Warren, G. Reakes and A. Massari, Business Rules Validation - the Standard the W3C Forgot. In *Proc. of the XML Europe 2003 Conference*, 2003, http://www.idealliance org/papers/dx_xnle03/papers/03-02-03/03-02-03.