プログラムテンプレートを用いた XML 文書変換手法に関する研究

2003MT008 馬場 敬　　2003MT041 小島 守道
指導教員　野呂 昌満

1　はじめに

現在，さまざまなデータが XML で表現される．XML は，テキスト形式でデータ構造を表現することができ，データ構造が人間に理解しやすい．データが特定の環境に依存しないなどの特徴をもつ．XSLT[1] などの文書変換技術があり普及している．

XSLT を用いて XML 文書処理するさいの問題点として，XSLT の処理記述が XML データ定義に依存することがあげられる．つまり，処理記述内に直接要素名や属性名を記述しなければならない．類似したデータ構造に対して同様の処理をするプログラムでも毎回同じプログラムを記述する必要がある．XSLT に処理記述中の要素名や属性名を抽象化する機能が存在しない．

本研究の目的は，プログラムテンプレート (以下，テンプレートと呼ぶ) を用いて再利用可能な XML 文書変換手法を提案することである．テンプレートは，スキーマの構造とその構造に対する典型的な処理をまとめ，処理記述中に出現する要素名を抽象化したものである．一つ一つの要素の構造と処理をまとめると，処理の再利用が困難になる．対象を，階層構造をもつレコード型の XML 文書とすると，類似した構造がよく現れ，その構造に対して同様の処理をおこなうことが多い．テンプレートにより，これらの構造と処理を抽象化でき，再利用することが可能となる．

本研究は以下のように進める．

- テンプレートの構文，意味を定義
- テンプレートの例を記述
- テンプレートを利用した XML 文書変換手法の再利用性の考察

2　関連研究

提案するテンプレートは，XSLT をベースに設計した．XSLT を選択した理由は，XSLT が XML 文書処理に広く使用されているからである．

2.1　XSLT

XSLT は，W3C により標準化された XML 文書の形式変換言語である．XSLT では変換規則を宣言的に記述するが，その記述がデータ定義に依存するので再利用ができない．すなわち，変換規則中に変換前あるいは変換後の要素名などが直接記述されるので，データ構造とその処理が同じで要素名だけが異なる場合でも変換規則の再利用が出来ない．

例として CD と論文のデータ構造の一部を図 1 に，CD と論文の XML 文書を HTML 形式に変換するプログラムを図 2 に示す．図 2 の XSLT プログラムでは変換規則は template 要素として表現される．変換対象となる要素名は match 属性として記述され，変換の処理は子要素で記述される．

この例では，処理記述中に name, country, aref, authref という要素名や属性名が直接書かれているので，CD と論文では処理がほとんど同じであるにもかかわらず，再利用することができない．

```
CD のデータ構造
<!ELEMENT artist (name,country)>
<!ATTLIST artist aid ID #IMPLIED>
<!ATTLIST artist aref IDREF #IMPLIED>
論文のデータ構造
<!ELEMENT author (name)>
<!ATTLIST author authid ID #IMPLIED>
<!ATTLIST author authref IDREF #IMPLIED>
```

図 1　データ構造

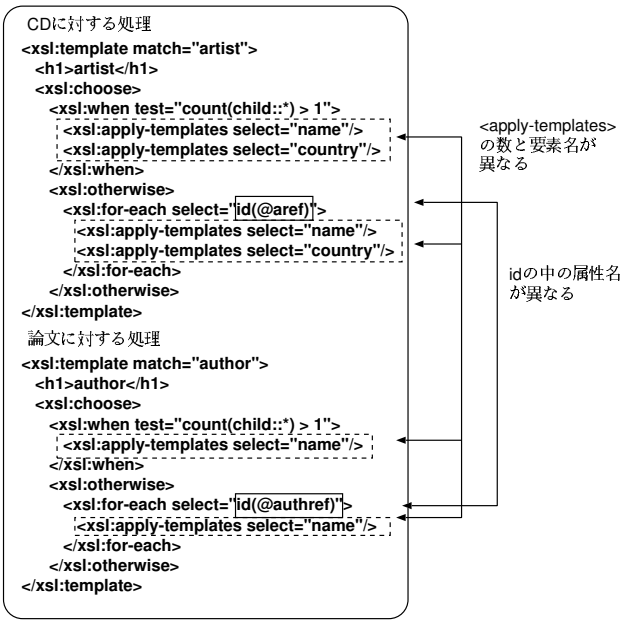

図 2　XSLT プログラムの例

3　テンプレートを用いた XML 文書変換手法

先に述べた問題点を解決するために，テンプレートを用いた XML 文書変換手法を提案する．テンプレートとは，スキーマの構造とその構造に対する典型的な処理をまとめ，その処理記述中に出現する要素名をパラメータで置き換え抽象化したものである．テンプレートは XML で記述され，テンプレート処理系により XSLT プ

ログラムに変換される．

3.1 テンプレートの設計指針

テンプレートは，スキーマのデータ構造とその構造に対する処理をカプセル化したものとした．テンプレートはオブジェクト指向のクラスとみなすこともできる．つまり，スキーマのデータ構造はクラスのフィールドに相当し，構造に対する処理はメソッドと考えられる．対象を階層構造をもつレコード型のデータとすると，スキーマのデータ構造からおこなう処理が決定できると考えた．また，データ構造もテンプレートとしてまとめたので，スキーマのデータ構造から適応できるテンプレートを容易に選択することができ，処理を決定することができる．

XML のデータ構造をクラスで表現する方法として，要素をクラスとし，子要素をクラスのメンバ変数としてあらわすことがある．JAXB[2] などのデータバインディングツールで用いられている．この方法では要素名はメンバ変数の変数名として表現されるので，要素名が異なるデータ定義ごとにクラス定義をしなければならない．われわれは変数名をメンバ変数の値として表現できるようにテンプレートを設計した．

3.2 テンプレートの定義

3.2.1 テンプレートの概要

テンプレートを用いたプログラムの全体の構造を図 3 に示す．テンプレートを用いたプログラムは，大きく分けて 2 つの部分に分けることができる．テンプレートを定義する TEMPLATE-DEFINITION と実際の XML 文書に対する XSLT の処理を記述する MAIN-SCRIPT である．TEMPLATE-DEFINITION ではテンプレートが定義されている．テンプレートはフィールド部とメソッド部に分けられる．MAIN-SCRIPT 部分では，従来と同じように XSLT プログラムを記述し，必要に応じてテンプレートを呼び出す処理を記述する．

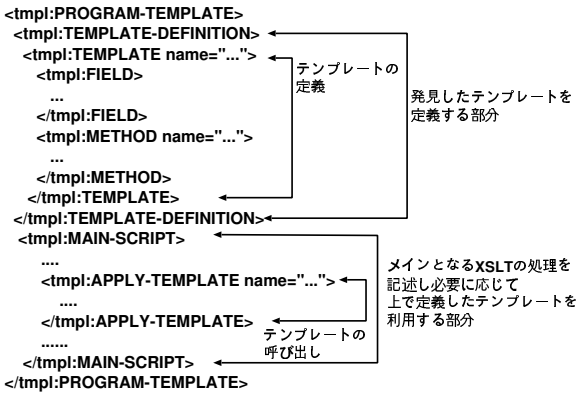

図 3　テンプレートを用いたプログラムの全体構造

3.2.2 フィールド定義

フィールド定義の例を図 4 に示す．フィールド部では，スキーマのデータ構造が変数により定義される．変数は XML の要素や属性を抽象化したものである．変数は，3.2.3 節で示す型を用いて定義する．変数の値は従来の XSLT プログラムで直接記述されていた要素名や属性名である．

```
<tmpl:FIELD>
  <tmpl:ELEMENT name="CHILD"/>
  <tmpl:ATTR name="LANG"/>
<tmpl:FIELD>
```

図 4　フィールド定義の例

3.2.3 型の分類

テンプレートで使用する型は以下のとおりである．

- 要素型 ELEMENT
- 属性型 ATTR
- テキスト型 TEXT
- リスト型 LIST
  - 要素型 ELEMENT
  - 属性型 ATTR
  - テキスト型 TEXT

要素型・属性型は XML 文書がもつ要素と属性に対応する型である．それぞれの変数値は，要素名・属性名となる．テキスト型は，属性値や XML のテキストを表現するために必要である．リスト型では同一の型の値を複数格納することができる．テンプレートでは，不定個の要素，属性やテキストが関連する処理が存在するので，リスト型が必要と考えた．XSLT 記述中に変数の値，つまり要素名や属性名を参照するときは，変数を % で囲んで使用する．% は，XSLT では用いられないので変数であることを示す記号に適していると考えた．

3.2.4 メソッドの定義

メソッド部ではフィールド部で表現されるデータ構造やメソッドの仮引数 (PARAMETERS 要素) に対する処理をあらわす．メソッド名は METHOD 要素の属性 name であらわされ，この名前を用いてメソッド呼出しがおこなわれる．メソッド本体の処理は BODY 要素内に 3.2.5 節で示す文を使用して記述される．メソッド定義の例を図 5 に示す．このメソッドは，名前が CHOICE_LANGUAGE であり，テキスト型のリスト LANGS が引数である．

```
<tmpl:METHOD name="CHOICE_LANGUAGE">
  <tmpl:PARAMETERS>
    <tmpl:LIST name="LANGS">
      <tmpl:TEXT/>
    </tmpl:LIST>
  </tmpl:PARAMETERS>
  <tmpl:BODY>
     テンプレートで定義した文
  </tmpl:BODY>
<tmpl:METHOD>
```

図 5　メソッド定義の例

3.2.5 文の定義

テンプレートで使用できる文は以下の 3 つである．

- XSLT の文
- テンプレートのメソッド呼出し
- FOR-EACH 文

XSLT の文では，要素名などは変数を用いて抽象化される．テンプレートのメソッド呼出しは，APPLY-TEMPLATE 要素の属性 name で，呼び出すテンプレートを指定する．そしてテンプレートのフィールド部やメソッドの引数に対応する値を指定しメソッドを呼び出す．図 6 は，図 5 で定義されたメソッドを呼び出す例である．メソッドの引数は”Ja” と”En” を要素とするテキスト型のリストである．

```
<tmpl:APPLY-TEMPLATE name="LANGUAGE">
  <tmpl:ELEMENT>name</tmpl:ELEMENT>
  <tmpl:ATTR>lang</tmpl:ATTR>
  <tmpl:METHOD call="CHOICE_LANGUAGE"/>
  <tmpl:LIST>
    <tmpl:TEXT>Ja</tmpl:TEXT>
    <tmpl:TEXT>En</tmpl:TEXT>
  </tmpl:LIST>
</tmpl:APPLY-TEMPLATE>
```

図 6　メソッド呼出しの例

FOR-EACH 文は，リスト型データの個々の値を処理するために用いる．図 7 に使用例を示す．

```
CHILDREN=("name","country","address") とすると，
<tmpl:FOR-EACH variable="CHILD" LIST="CHILDREN">
  <tmpl:BODY>
    <xsl:apply-templates select="%CHILD%"/>
  </tmpl:BODY>
</tmpl:FOR-EACH>
              ↓
<xsl:apply-templates select="name"/>
<xsl:apply-templates select="country"/>
<xsl:apply-templates select="address"/>
```

図 7　FOR-EACH 文の例

テンプレートでは繰り返しをおこなう FOR-EACH 文が存在するが，IF 文はテンプレートの構文として定義しなかった．理由は XSLT 自体の構文で，分岐処理をおこなう if 文や choose 文が用意されているので，それらを利用して分岐処理を記述できる．

3.2.6　実現したテンプレート処理系

今回実現したテンプレート処理系を用いた処理の流れを図 8 に示す．Java を用いて DOM[3] 木を生成して実現した．

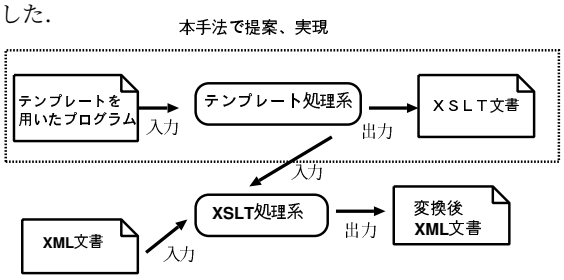

図 8　テンプレートを用いた XML 文書変換の流れ

3.3　テンプレートの例

われわれが定義したテンプレートを示す．よく出現するスキーマ構造からテンプレートを定義した．

3.3.1　言語テンプレート

データ構造

人物や会社などの名前や論文の題名などの複数の言語 (英語，日本語など) で表現されるデータをもつ構造である．

XML **による表現**

データは要素であらわされ，データの記述言語は属性であらわされる．

よく用いられる処理

言語に優先順位をつけて表示するメソッド CHOICE_LANGUAGE，指定した言語表現をすべて出力するメソッド PRINT_LANGUAGE が考えられる．優先順位はリストで表現する．テンプレート定義を図 9 に示す．PRINT_LANGUAGE の内容は省略した．

```
<tmpl:TEMPLATE name="LANGUAGE">
  <tmpl:FIELD>
    <tmpl:ELEMENT name="CHILD"/>
    <tmpl:ATTR name="LANG_ATTR"/>
  </tmpl:FIELD>
  <tmpl:METHOD name="CHOICE_LANGUAGE">
    <tmpl:PARAMETERS>
      <tmpl:LIST name="LANGS">
        <tmpl:TEXT/>
      </tmpl:LIST>
    </tmpl:PARAMETERS>
    <tmpl:BODY>
      <xsl:choose>
        <tmpl:FOR-EACH variable="LANG" list="LANGS">
          <tmpl:BODY>
            <xsl:when test
                 ="%CHILD%[%LANG_ATTR%=%LANG%]">
              <xsl:apply-templates select
                   ="%CHILD%[%LANG_ATTR%=%LANG%]"/>
            </xsl:when>
          </tmpl:BODY>
        </tmpl:FOR-EACH>
      </xsl:choose>
    </tmpl:BODY>
  </tmpl:METHOD>
  <tmpl:METHOD name="PRINT_LANGUAGE">
      ・・・
    </tmpl:METHOD>
</tmpl:TEMPLATE>
```

図 9　言語テンプレート

3.3.2　定義参照テンプレート

データ構造

複数の箇所で共通に使用されるデータが存在する場合に，一ヶ所でデータの詳細を定義し，その定義に名前をつける．他の箇所からは名前を指定してそのデータを参照する．

XML **による表現**

定義箇所は ID 属性をもつ要素としてあらわす．参照箇所は IDREF 属性をもつ要素であらわす．定義要素，参照要素は同じ要素名とする．

よく用いられる処理

GET_REFS は，IDREF 属性が参照している要素を取得して処理をおこなう．IDREF 属性が存在しなければ子要素を取得する．メソッド GET_CHILDREN は，子要素が存在すれば子要素を取得し，子要素が存在しなければ参照先の要素を取得する．テンプレート定義を図 10 に示す．メソッド GET_REFS の処理内容を省略した．

```
<tmpl:TEMPLATE name="IDIDREF">
  <tmpl:FIELD>
    <tmpl:LIST name="CHILDREN">
      <tmpl:ELEMENT/>
    </tmpl:LIST>
    <tmpl:ATTR name="REFNAME"/>
  </tmpl:FIELD>
  <tmpl:METHOD name="GET_CHILDREN">
    <tmpl:PARAMETERS/>
    <tmpl:BODY>
      <xsl:choose>
        <xsl:when test="count(child::*) >=1">
          <tmpl:FOR-EACH
                variable="CHILD" LIST="CHILDREN">
            <tmpl:BODY>
              <xsl:apply-templates
                              select="%CHILD%"/>
            </tmpl:BODY>
          </tmpl:FOR-EACH>
        </xsl:when>
        <xsl:otherwise>
          <xsl:for-each select="@id(%REFNAME%)">
           <tmpl:FOR-EACH
                variable="CHILD" LIST="CHILDREN">
             <tmpl:BODY>
               <xsl:apply-templates
                               select="%CHILD%"/>
             </tmpl:BODY>
           </tmpl:FOR-EACH>
          </xsl:for-each>
        </xsl:otherwise>
      </xsl:choose>
    </tmpl:BODY>
  </tmpl:METHOD>
  <tmpl:METHOD name="GET_REFS">
    ・・・
  </tmpl:METHOD>
</tmpl:TEMPLATE>
```

図 10　定義参照テンプレート

4　考察

本研究で提案する XML 文書変換手法の処理の再利用性，テンプレート機能の拡張，一般のプログラミング言語との比較について考察する．

4.1　処理の再利用性

われわれはテンプレートの定義をよく出現するスキーマ構造を単位におこなった．処理記述が再利用できる例を図 11 に示す．このプログラムは，図 10 のテンプレートを呼び出し，処理すると図 2 の XSLT プログラムが生成できる．テンプレートの名前や出現する変数名は，要素名や属性名とは無関係である．変数の値に実際の要素名や属性名が格納される．

更なる再利用性の向上の方法として，テンプレート定義のライブラリ化があげられる．テンプレート定義のライブラリ化とは，テンプレートを用いたプログラムの TEMPLATE-DEFINITION と MAIN-SCRIPT を別のファイルに記述することである．

4.2　テンプレートの拡張

テンプレートの拡張として考えられるのは，ライブラリへのメソッドの追加と型チェックの追加である．ライブラリへのメソッドの追加とは，ライブラリで定義されている既存のテンプレートに対して，ライブラリのファイルを操作すること無くメソッドを追加できる機能のことである．オブジェクト指向のクラス継承と同様に処理をおこなえばよいと考えられる．また，現在処理系では型チェックをおこなっていないが，型チェックを追加すればユーザの誤りを早期に検出できる．

```
CD データに対する記述
<xsl:template match="artist">
  <tmpl:APPLY-TEMPLATE name="IDIDREF">
    <tmpl:LIST>
      <tmpl:ELEMENT>name</tmpl:ELEMENT>
      <tmpl:ELEMENT>country</tmpl:ELEMENT>
    </tmpl:LIST>
    <tmpl:ATTR>aref</tmpl:ATTR>
    <tmpl:METHOD call="GET_CHILDREN">
  </tmpl:APPLY-TEMPLATE>
<xsl:template>
論文データに対する記述
<xsl:template match="author">
  <tmpl:APPLY-TEMPLATE name="IDIDREF">
    <tmpl:LIST>
      <tmpl:ELEMENT>name</tmpl:ELEMENT>
    </tmpl:LIST>
    <tmpl:ATTR>authref</tmpl:ATTR>
    <tmpl:METHOD call="GET_CHILDREN"/>
  </tmpl:APPLY-TEMPLATE>
</xsl:template>
```

図 11　テンプレートの再利用性

4.3　プログラミング言語との比較

XML はデータ構造の記述に特化しており，類似したデータ構造や定型処理が多く出現し，再利用の問題が顕在化する．

この問題は一般のプログラミング言語でも存在する．プログラミング言語ではデータ構造はレコード型などにより定義される．プログラムごとにレコードのメンバ名が異なる場合が多く，類似したレコード型に対する処理は再利用されていない．しかし，一般のプログラムでは類似したデータ構造でもそれに対する処理が多岐にわたることが多いので，ある特定の処理における再利用性の低さが顕在化しなかったと言える．

5　おわりに

本研究では，テンプレートを利用して XML 文書に対する処理を再利用する手法を提案した．テンプレートに必要な型，構文の整理をおこなった．次にテンプレートの記法を定義し，テンプレートの抽出をおこなった．最後に提案した変換手法の再利用性や拡張について考察をおこなった．

今後の課題としてテンプレート記法の変更，テンプレートの例を充実させることがあげられる．本研究では，テンプレートを XML で記述した．しかし，記述が冗長になったのでより短い記法を提案する必要がある．

謝辞

本研究を進めるにあたり，熱心な御指導をいただいた野呂昌満先生，有益なアドバイスをくださった張漢明先生，蜂巣吉成先生，野呂研究室，張研究室のみなさまに感謝いたします。

参考文献

[1] James Clark, XSL Transformations(XSLT), http://www.w3.org/TR/xslt，1999.

[2] Sun, Java Architecture for XML Binding (JAXB) http://java.sun.com/webservices/jaxb/,

[3] W3C, Document Object Model (DOM), http://www.w3.org/DOM/，1997.