ソフトウェア工学　139－9
(2002. 10. 11)

情報交換のためのXML変換言語XTLとXSLTとの比較検討

梅崎 利矢　横関 大子郎　村山 隆彦
日本電信電話株式会社 NTT情報流通プラットフォーム研究所
〒180-8585 東京都武蔵野市緑町3-9-11
E-mail: {umezaki.toshiya, yokozeki.daigoro, murayama.takahiko}@lab.ntt.co.jp

あらまし
企業間の情報共有を効率化し，製品開発のリードタイムを短縮するための方法として，各企業の帳票文書をXMLドキュメントとして作成し，交換することが考えられる。本論文では，入力帳票と出力帳票のXML文書の要素を制約規則に基づいて対応付けておき，この制約規則に基づいてXML文書の操作を実行するXML変換言語XTLを提案し，XSLTとXTLの制約規則を要素レベルで比較することにより，XTLの記述容易性を明らかにした。

キーワード
XML, 変換, XSLT, 帳票

A Study of XML Transformation Language XTL and XSLT

Toshiya UMEZAKI　Daigoro YOKOZEKI　and　Takahiko MURAYAMA
NTT Information Sharing Platform Laboratories, NTT Corporation
3-9-11 Midoricho Musashino-shi Tokyo, 180-8585 Japan
E-mail: {umezaki.toshiya, yokozeki.daigoro, murayama.takahiko}@lab.ntt.co.jp

Abstract
Creating and exchanging business documents by XML format has proved its efficiency in information exchange and reduced the lead time in product development.
In this article, we will propose XTL, the XML transformation language, which maps input/output XML under context rules and execute the program. We will clarify its simple scripting by comparing XSLT and XTL.

Keyword
XML, transform, XSLT, form

1. はじめに

インターネットの普及により電子商取引市場が急速に立ち上がってきている。企業間の情報共有を効率化し，製品開発のリードタイムを短縮するための方法として，各企業の帳票文書を XML ドキュメントとして作成し，交換することが考えられる[1]。製品開発にかかわる伝票や仕様書等は階層的な構造をもつため XML で容易に表現できる。

XML を操作するための方法として XSLT[2]が提案されている。しかし XSLT は入力としての XML 文書を段階的に構文解析しながら出力としての XML 文書への変換を記述するため，変換の定義が手続き的で，生成される XML 文書の構造が XSLT の記述から容易に判読できないだけでなく，業務担当者では記述が困難であることが問題であった。さらにアプリケーション開発者による変換処理の開発が必要になるという点も問題であった。

この問題を解決するため，入力側と出力側の XML 文書の要素を制約規則に基づいて対応付けておき，この制約規則に基づいて XML 文書の操作を実行する XML 変換言語 XTL[2]を提案した。

本論文では，XSLT と XTL の制約規則を要素分解し，各要素を種類別に分類することにより，XTL の記述容易性を明らかにする。

2. XTL 概要

XTL の適用ターゲットは企業間で交換される伝票や仕様書等の帳票である。ここでの帳票とは構造を持つ値の集合を意味し，XML ドキュメントとして表現されているものを指す。

XTL は，業務担当者が容易に文書間の依存関係を定義できるよう，転記・選択・同値分割といった変換操作と，入力帳票と出力帳票の対応関係を表す制約規則とに分離している。よって，同じ制約規則であっても，解釈する処理エンジン(XTE)によって異なる処理を行うことができる。XTL の構文については，付録に掲載する。なお，XTL 自身も XML で表現されている。

図 1 に入力帳票から出力帳票へ，構造変換を伴わない転記の例を示す。構造変換を伴わない転記とは，入力帳票と出力帳票の構造が同じ[1]であり，かつタグに囲まれている値[2]も同じまま，タグ名のみを変更する処理である。

図 1 の転記を行う場合に記述する XTL は図 2 になる。<from>～</from>内に入力帳票の構造を，<to>～</to>内に出力帳票の構造を埋め込む。値は *1, *a, ?1, ?a 等の変数として扱い，入力帳票と出力帳票での対応関係を示している。

図 2 では図 1 のパターンを処理するため，入力帳票と出力帳票の構造を全て埋め込んでいるが，必要に応じて，変換処理の対象となる部分のみを記述することもできる。

通常，この制約規則を解釈する処理エンジンは，<from>内の条件に合わせて収集した値を<to>内の条件[3]に合わせて展開する。ここで，<from>と<to>は対称的な構造を持っているため，値の操作や同値分割を含まない制約規則であれば，逆方向に制約規則を適用する操作を与えることにより，逆変換への対応も可能である。

図 1 の変換処理を XSLT で記述すると図 3 のようになる。規則の上の行から順番に入力帳票中の値を構文解析していき，出力帳票の構造に合わせて段階的に埋め込んでいく，手続き指向の書式になっている。

規則というよりは処理手順が記述されているため，逆変換や類似の変換を行う際にも，新しい規則を書き起こす必要がある。

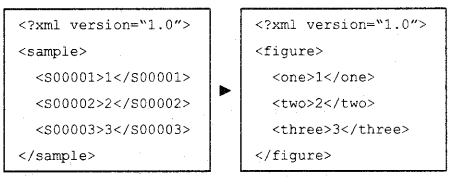

図 1: 変換例-構造変換のない転記

```
<?xml version="1.0"?>
<rule>
  <section name="XXX">
    <from>
      <sample>
        <S00001>*1</S00001>
        <S00002>*2</S00002>
        <S00003>*3</S00003>
      </sample>
    </from>
    <to>
      <figure>
        <one>*1</one>
        <two>*2</two>
        <three>*3</three>
      </figure>
    </to>
  </section>
</rule>
```

図 2: XTL

[1] 入力帳票と出力帳票の様式が同じで，タグ名(欄名)のみが異なるイメージ。タグ名も含めてまったく同じ場合の転記は，複写と呼んでいる。

[2] 帳票で言うところの記入欄内に記入された値に該当する。

[3] この例では構造条件のみであるが，数式やマクロを記述することもできる。

```
<?xml version="1.0">
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="sample">
  <figure>
    <xsl:apply-templates/>
  </figure>
</xsl:template>
<xsl:template match="S00001">
  <one>
    <xsl:apply-templates/>
  </one>
</xsl:template>

<xsl:template match="S00002">
  <two>
    <xsl:apply-templates/>
  </two>
</xsl:template>
<xsl:template match="S00003">
  <three>
    <xsl:apply-templates/>
  </three>
</xsl:template>
</xsl:styleshee>
```

図 3: XSLT

3. 比較検討方法

帳票に対して必要な操作[3]のうち，代表的な 3 操作である同値分割・転記・選択について XTL と XSLT の記述容易性に関する比較を行う。

まず，帳票の操作について述べる。

同値分割とは，入力帳票中のキーとなる値の同値関係に基づき，入力帳票を分割する処理である。

転記とは，入力帳票中の値を出力帳票の構造に合わせて書き写す処理である。タグの親子関係が逆転するなどの構造変換がなく，タグ名のみ変換する処理と，構造変換を伴う処理の 2 通りがある。

選択とは，入力帳票から値と構造を合わせた形で一部分を抜き出す処理である。

次に，比較の方法について述べる。

比較は，要素を種別ごとに分けて行う。ただし，文頭の 1 行にある XML 宣言は対象に含まない。

ここで要素とは，タグ名(<A>sample</A>内の「<A>」「</A>」)・タグ値(<A>sample</A>内の「sample」)・属性名(<A option="ALL">内の「option」)・属性値(<A option="ALL">内の「"ALL"」)の 4 つからなる。

タグ名は，開き(<A>)と閉じ(</A>)を合わせて 1 要素とする[4]。namespace 表現は，その後に続くタグ名とペアで考え，単独ではカウントしない。

例えば <xsl:template match="test"> という行は，xsl:template, match, "test" の 3 要素に分割できる。<A>name</A> という行は，<A></A>, name の 2 要素である。

種別は，固定要素・入力該当要素・出力該当要素・処理該当要素の 4 種である。

固定要素とは，rule, section, xsl:stylesheet といった言語仕様として規定されている予約語相当の記述である。

入力該当要素とは，入力文書中のタグ名や値等，入力文書に出現する要素の記述である。

出力該当要素とは，出力文書中のタグ名や値等，出力文書に出現する要素の記述である。

処理該当要素とは，*1, match 等，値の収集に使う記述や条件文である。

この時，例えば図 2 の場合は以下のように分類できる。

- 固定要素
  - rule, section, name, "XXX", from, to の 6 つ
- 入力該当要素
  - sample, S00001, S00002, S00003 の 4 つ
- 出力該当要素
  - figure, one, two, three の 4 つ
- 処理該当要素
  - *1, *2, *3 (各 2 つずつ) の 6 つ

同様に図 3 の場合は，以下のように分類できる。

- 固定要素
  - xsl:stylesheet, xmlns:xsl, "http://www.w3.org/1999/XSL/Transform" の 3 つ
- 入力該当要素
  - sample, S00001, S00002, S00003 の 4 つ
- 出力該当要素
  - figure, one, two, three の 4 つ
- 処理該当要素
  - xsl:template, match, apply-templates (各 4 つずつ)の 12 個

固定要素は予め言語仕様で決められているため，機械的に付加および削除することができる。また，入力該当要素・出力該当要素は，入力帳票・出力帳票から抜き出して流用することができる。

よって，残りの処理該当要素数が変換に要する純粋な記述コストと考えることができる。処理該当要素数が少ない制約規則ほど入力帳票と出力帳票から類推しやすく，記述が容易であるといえる。

4. 比較検討結果

今回の検討で使用した入力帳票および出力帳票のスペックは以下のとおりである。このうち，<転記(構造変換なし)> は図 1 に示しているものである。また，<転記(構造変換あり)>は図 4 に示

[4] <A/>も 1 要素である。

しているものである。

なお，図 4 の変換を XTL で記述したものが図 5，XSLT で記述したものが図 6 である。

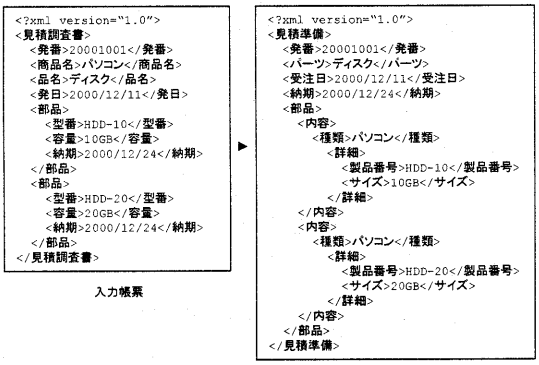

図 4: 変換例(2) - 構造変換のある転記

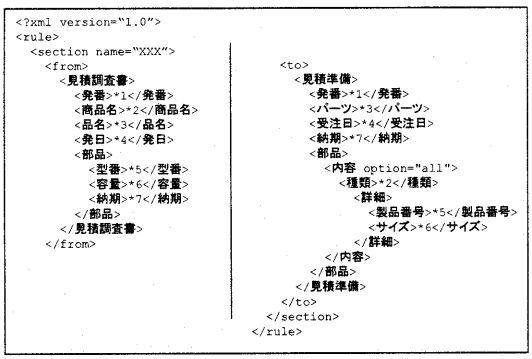

図 5: XTL(2)

<同値分割>

- 入力帳票: 30 要素, 3 階層, 4 タグ組(2 階層)×3 回の繰り返しあり，分割のキーとなる値 2
- 出力帳票 1: 23 要素，3 階層，4 タグ組(2 階層)×2 回の繰り返しあり，構造変換あり
- 出力帳票 2: 16 要素, 3 階層，繰り返しなし，構造変換あり

<転記(構造変換なし)>

- 入力帳票: 7 要素, 2 階層，繰り返しなし
- 出力帳票: 7 要素, 2 階層，繰り返しなし

<転記(構造変換あり)>

- 入力帳票: 23 要素, 3 階層，繰り返しなし
- 出力帳票: 27 要素, 5 階層, 5 タグ組(2 階層)×2 回の繰り返しあり

<選択>

- 入力帳票: 11 要素, 3 階層, 3 タグ組(2 階層)×2 回の繰り返しあり
- 出力帳票: 3 要素, 2 階層，繰り返しなし，構造変換あり

以上の各処理について XTL と XSLT の制約規則を用意し，3.節で述べた方法にしたがって要素分類した。変換処理パターンごとの各要素数をまとめたものが図 7 である。

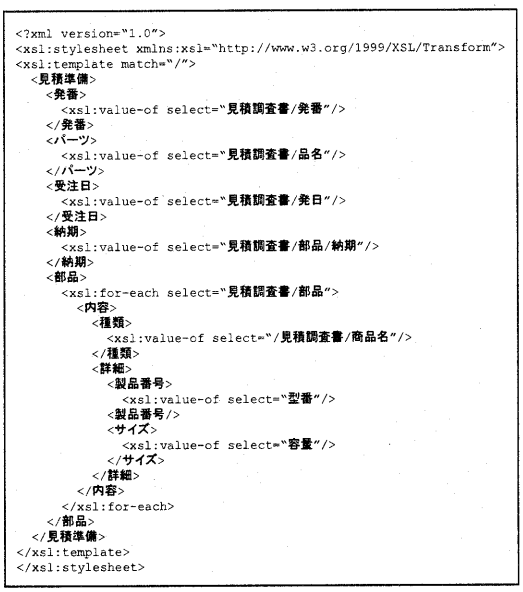

図 6: XSLT(2)

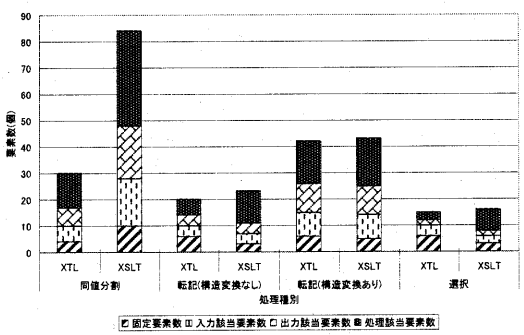

図 7: 分類結果グラフ

5. 考察

前節の結果より，次の 2 点が確認された。

- XSLT と比較して，XTL の記述が容易であること
- 入力帳票のタグ数が増加するに従い，XTL の記述優位性が高まること

以下，それぞれの詳細について述べる。まず，要素種別ごとに特徴を見る。

XTL の総要素数と処理該当要素数が XSLT に比べて全般的に少ないことがわかる。これは，XTL 全体の記述量が少なく，かつ変換に直接関係する処理該当要素数が少ないことから，記述が容易であることを表している。

入力該当要素・出力該当要素の数では同値分割を除いて両者に大きな差は見られない。これは，実験に用いた入出力帳票のタグ数が少なかったせいだと推測できる。詳細は後述する。

同値分割に関して XSLT の入力該当要素数・出力該当要素数が増えているのは，出力する帳票数と同数の制約規則が必要なためである。前節の条件では，入力帳票中にキーとなる値が 2 つあるため，出力帳票が 2 つになる。XSLT は，1 出力帳票について 1 規則必要であるため，要素数は全て 2 倍になっている。

一方，XTL は出力帳票数には比例せず，常に 1 つの制約規則でよい。また入力帳票中のキーとなる値の数が未定，つまり出力帳票数の数が予め定まっていない場合でも処理可能であり，XSLT より有利である。

次に，変換処理ごとに特徴を見る。

同値分割では，総要素量・処理該当要素数とも明らかに XTL の記述量が有利であることがわかる。

構造変換なしの転記および選択処理においては総要素数に大きな差がないものの，処理該当要素数を比較すると XTL が有利である。

構造変換を伴う転記においては，大きな差は見られない。

ここで，入力帳票中のタグ数が増えた場合について考察する。構造変換なしの転記において，入力帳票内のタグ数を n とすると，要素数は以下の式で表せる。

<XTL>
- 合計要素数: 4n+8
  - 固定要素数: 6
  - 入力該当要素数: n+1
  - 出力該当要素数: n+1
  - 処理該当要素数: 2n

<XSLT>
- 合計要素数: 5n+8
  - 固定要素数: 3
  - 入力該当要素数: n+1
  - 出力該当要素数: n+1
  - 処理該当要素数: 3n+3

これは，入力帳票中のタグ数が増加した場合に XTL のほうが有利であることを示している。

6. おわりに

企業間情報交換に XML を用いることを想定した場合に，XTL と XSLT それぞれの規則を要素分解することによって，XTL 内の変換に直接要する処理該当要素数が少なく，記述が容易であることを明らかにした。また，入力帳票のタグ数が増えるに従い，XTL の記述優位性が高まることもわかった。

今後は，XTL の記述範囲を定式化することにより，適用範囲を明確にし，XML Query[4]等の多言語へ変換する方法や制約規則の自動生成法[5]についても議論を進めていきたい。

A. 付録

XTL の構文を以下に示す。ここで，<>で囲まれた英文字列で制約ルールの予約語タグを示す。また，a{,a}...で構文要素の繰り返しを表す。[a]で a があるかないかのいずれかであることを示す。a | b で構文要素 a と b のいずれかの選択を示す。英字列は非終端記号である。

XTL の制約規則は複数のセクションから構成される。各セクションは，ソース・テンプレートとターゲット・テンプレートから構成され，一組の対応関係を表現している。

```
Rule ::= <rule> Section {, Section}... </rule>
Section ::= <section>
        SourceTemplate
        TargetTemplate
        </section>
SourceTemplate ::= <from> Template </from>
TargetTemplate   ::= <to> Template </to>
```

変換操作の場合，ソース・テンプレートが XML 文書の問い合わせにおける値抽出検索部分を表現し，ターゲット・テンプレートが XML 文書の

再構築部分を表現している。

テンプレートはテンプレート要素の入れ子からなる階層構造を持つXMLデータである。

```
Template ::=
        <TagName>TemplateBody</TagName>
TemplateBody ::=
        <TagName> TemplateBody </TagName>
{, <TagName> TemplateBody </TagName> }...
| TemplateElement
```

テンプレート要素は，以下のいずれかである。

- 通常のXMLデータ要素 XMLelement
- 条件式 Expression
- 転記型タグ値変数 CopyTagValue
- 類別型タグ値変数 ClassTagValue
- 生成型タグ値変数 GenerateTagValue
- マクロタグ値名 MacroTagName

```
TemplateElement ::=
        XMLelement | Expression
      | CopyTagValue | ClassTagValue
      | GenerateTagValue
      | %MacroTagName%
CopyTagValue ::=
        *TagValueName [ Expression ]
ClassTagValue ::=
        ?TagValueName [ Expression ]
```

※Expression は既知とする。

文　献

[1] 唐沢裕明, "VMに向けたプラットフォームサービス－繊維アパレル業界プラットフォーム「ApparelArc」", NTT 技術ジャーナル, Vol.12, No.7, pp.28-31, 2000.

[2] XSLT, http://www.w3.org/TR/xslt

[3] 横関大子郎, 村山隆彦, 山本修一郎, "制約規則に基づく XML 情報変換方式の提案", 信学技報, Vol.100, No.206, pp.83-90, 2000.

[4] XML-Query, http://www.w3.org/XML/Query.html

[5] 梅崎利矢, 村山隆彦, "XML を用いた構造データ交換方法に関する一考察", 第63回情処全国大会, 5W-4.