分散 XML 言語処理系の設計

佐藤 義弘† 佐藤 聡‡ 板野 肯三‡

†筑波大学情報学類 〒305-8673 茨城県つくば市天王台 1-1-1
‡筑波大学電子・情報工学系 〒305-8573 茨城県つくば市天王台 1-1-1
E-mail: †yoshihiro@hlla.is.tsukuba.ac.jp, ‡{akira, itano}@is.tsukuba.ac.jp

あらまし 近年 XML 形式のデータの流通量は増加してきている．また，XML に関する様々な技術が提案されている. 例えば, Web サービスは各プロバイダが管理している知的情報の XML 形式によるデータ配信の技術であり，XSLT は XML 形式のデータの編集統合の技術である．今後は，これらの技術を組み合わせ，付加価値を与えた知的情報の発信が盛んに行われるようになると思われる．しかしながら，現在の段階では，これらの技術を組み合わせたサービスの提供を容易に実現できるフレームワークが存在していない．本研究では，インターネット上に分散する XML 配信サービスや，XML 編集サービスを柔軟に結合し，付加価値を与えた新しいサービスを容易に実現するためのメタフレームワークの提案とその実現を行う

キーワード XML，言語処理系，情報統合，問い合わせ処理，情報検索

A designing of distributed XML processing language system

Yoshihiro SATOU† Akira SATO‡ and Kouzou ITANO‡

†College of Information Sciences, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573 Japan
‡Institute of Information Sciences and Electronics, University of Tsukuba
1-1-1 Tennodai, Tsukuba, Ibaraki, 305-8573 Japan
E-mail: †yoshihiro@hlla.is.tsukuba.ac.jp, ‡{akira, itano}@is.tsukuba.ac.jp

Abstract In recent years, a flow of XML format data is increasing. Also some technologies for XML are proposed. For example, Web service is a data delivery technology by XML format of intellective information administrated by each providers, XSLT is a technology for editing and integrating of XML format data. In the future, information delivery that combines these technologies and gives added value is thought to become active. However, at present, there is no framework that facilitates providing a service to combine these technologies. This research proposes a meta-framework that integrates flexibly XML delivery services and XML editing services distributed on Internet, and realizes it.

Keyword XML, Language Processing System, Information Integrating, Query Processing, Information Search

1. はじめに

XML[1]はここ数年で急速に成熟している技術である．そのため近年では XML 形式のデータの流通量は年々増加し，また XML の有用性から，例えば各プロバイダが管理している知的情報の XML 形式によるデータ配信の技術や，XSLT による XML 形式データの編集，統合など，XML に関する様々な技術が提案されている．

これら XML に関する技術は今後もますます増えてくるであろうし，またこれらの技術を組み合わせ，付加価値を与えた知的情報の発信は，今後盛んに行われるようになると予想される．例えば「入力された郵便番号を住所に変換する」という XML 編集サービスと「入力された住所から大まかな緯度,経度を出力する」という XML 配信サービスを結合して「入力された郵便番号から大まかな緯度，経度を出力する」という XML 配信サービスを新たに提供する, といった具合である．

しかし，現在の段階ではこれらの XML に関する技術を組み合わせたサービスや，付加価値を与えるサービスを容易に提供できるメタ的なフレームワークは存在しておらず,それらのサービスを提供するためには，サービスの提供者が独自の手法によってそれらの技術を組み合わせたり付加価値を与えたりする必要があり，それらのサービスの提供は提供者にとって大きな負担となっている．またメタ的なフレームワークが存在したとすれば，サービスの利用者としても，いくつかのサービスを組み合わせることで必要とする情報を得る

ことができるようになると考えられる．

そこで本研究では，これら Web 上に分散する XML 配信サービスや XML 編集サービスを柔軟に結合し，付加価値を与えるような新しいサービスの提供を容易に実現するためのメタフレームワークを提案する．

2. 分散 XML 処理言語系の設計，および実装

この章では設計した言語，およびその処理系について説明する．

本研究で提案するメタフレームワークは, XML 文書を入力として受け取り，それを処理した結果を XML 文書として出力するためのものである．本研究ではこのフレームワークを実現するために必要な処理言語，およびその言語を解析して処理を行う処理系の設計，開発を行う．

2.1. 処理言語に求められる機能

本研究では，インターネット環境のように，多数の情報源がネットワークによりアクセスできる環境を対象としている．このとき，利用者が行うデータ処理としては, 複数の情報ソースから必要な情報を抜き出し，加工し，それらを１つに纏めることで最終的にユーザが必要としている情報を作り出す，という処理が一般的である．この処理過程を図 1に示す．

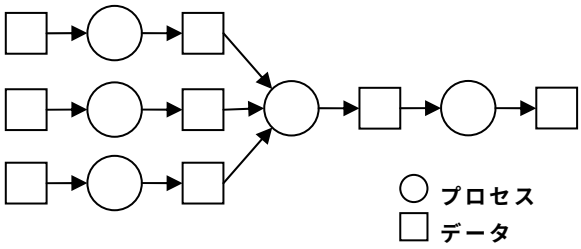

図 1　処理例

図 1は３種の情報源から必要な情報を加工して取り出し，それらを１つのデータとして纏め，さらに纏めた情報を加工して最終的に必要とされる情報を構成する，というプロセスを示している．従って，データ処理を記述する言語には，１つのプロセスを節とした木構造となる処理手順を記述できなければならない．

図 1の例から分かるように，このようなデータ処理はプロセスを節とした木構造を成している．そのため木構造の言語である XML との親和性に優れていると考えられる．そこで本研究では処理手順を記述する言語として，木構造の特性を持つ XML 形式の言語が有効であると考えた．また，本研究の目的である Web 上に分散する XML 関連サービスを統合したり，付加価値を付与したりするための処理言語を XML 形式言語とすることで，本フレームワークの利用者の XML に関する知識，情報をそのまま活用することができる，という副次的な効果も見込まれると考える．

また，Web 上のデータへのアクセスという観点からも XML は有効である．Web 上のデータソースとしては HTTP GET のような Web 上にあるデータをそのままの形で利用者に提供する静的なデータソース，および HTTP POST や SOAP などに見られる, Web 上のデータに対して何らかの処理を施しその結果を利用者に提供する動的なデータソースの大きく２つのデータソースが考えられる．これらのデータソースへアクセスするためにはデータへのアクセスのための手順の定義，即ちインタフェースの定義が必要となる．この Web 上のデータへのインタフェース定義に用いられるものとして WSDL[2]がある．この WSDL は XML 形式の言語であり，XML との親和性に大変優れている点で XML 形式の言語が有効であると言える．

同様にデータの加工の面からも XML の選択を促す点がある. XML 文書を加工するための最も簡単な手法に XSLT[3]エンジンを用いる方法がある. XSLT はソースとなる XML 文書に何かしらの変換を施し，変換した XML 文書を出力するために用いられるスクリプト言語である. この XSLT 自身も XML 形式の言語として定義されており, XML の利用者であれば容易に習得し利用することが可能である．

これらの点を踏まえ，本研究では XML 関連サービスを統合し，付加価値を与え XML 文書を操作するための処理言語として XML 形式の言語を採用することとした．

2.2.　分散 XML 処理言語系の概要

本研究で設計，開発する言語と処理系の関係を図 2に示す．図 2中上部の四角の中の網掛け部分は本研究で設計した言語に当たる部分を，下の四角の中の網掛け部分はその言語を処理系で処理した結果である部分を示す．処理系は入力 XML 文書中の本言語で記述された部分を読み取って処理し，入力 XML 文書中の本源後で書かれた部分とその処理結果を部分と置き換えた文書を出力 XML 文書として出力する．

本研究で提案する処理言語系では XML 文書に対する処理の過程において, XML 文書の各要素を関数のように処理する，所謂関数型言語の概念を取り入れている．これは XML 文書がツリー構造で表すことが可能であることに着目したものである. XML 文書をツリー構造で表した例を図 3に示す．またこの例を関数式で

表した例を式(1)に示す．

$$f(g(x),h(p(y))) \quad \cdots\cdots \text{式(1)}$$

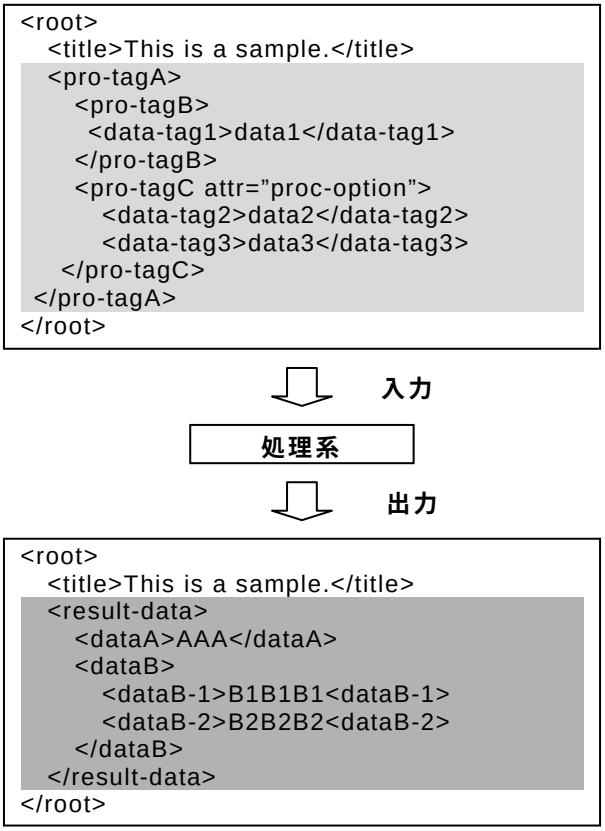

図 2 提案する処理言語系の概要

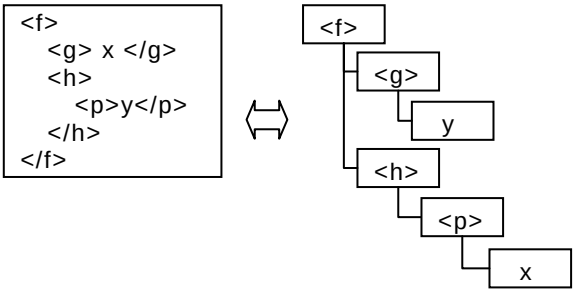

図 3 XML 文書と木構造

このとき，XML 文書の各ノードを処理命令，即ち関数として捉える．各処理はその関数の引数，即ち関数を表すノードの子ノード群である部分木（部分文書）を引数として適用する．式(1)に示す関数式の例において，関数 f の処理は<f>要素の子ノード群を表す部分木<g>（関数式 $g(x)$），および<h>（関数式 $h(p(y))$）を引数として適用される．このとき，ある関数を評価する際に，その関数の引数を関数の組み合わせと見て先に評価する（すなわち部分関数式の処理結果を引数とする）か，または，引数を単なる記号列の並びと見なして関数式としては評価しないか，という選択が考えられる．式(1)に示す関数式の例において，関数 f の処理に先行して，その引数である部分木 $h(p(y))$ を関数式と見なし先に関数 h を評価（処理）した結果を実引数とするのか，または，部分木 $h(p(y))$ は単なる記号列であると見なし，”<h> y </h>” という記号列を実引数とするのか，のいずれかの方法が考えられる．本言語処理系ではこれらの違いを関数の引数に対する eager evaluation（先行評価），lazy evaluation（遅延評価）の違いとして捉え，この言語の利用者が式中のすべての関数に個別に評価のタイミングを指定できるようにしている．

また，本フレームワークは Web 上に分散する XML 配信サービスや XML 編集サービスなどのサービスを統合し，付加価値を与えるために利用することを目的としている．本言語処理系も１つの Web サービスとすることにより，本フレームワークから本フレームワークを再帰的に利用可能となる．これにより，利用者が記述した一連のデータ処理プロセスを分散処理することが可能となる．また，評価結果となる XML 文書を処理手順とみなして，さらに処理を施すといったより高度な付加価を与えることを可能とする．

2.3. 言語仕様

ここではメタフレームワークのために設計した言語の仕様について述べる．

本研究で提供する処理言語は XML 形式の言語で，XML 文書を入力として受け取り XML 文書を結果として出力するためのものである．

入力された XML 文書から出力する XML 文書を構成するためには大きく２つの方法が考えられる．１つは入力の XML 文書のある部分木（XML ノード）の構成を規則に基づいて取り替えるといった部分木の更新処理で，もう１つは入力の XML 文書に全く新しい別の部分木を生成し，生成した部分木を入力の XML 文書に挿入する部分木の新規作成処理である．

本フレームワークでは部分木の更新処理については XSLT エンジンを用い，部分木の新規作成処理については Web サービスを用いて行うことを考え，本言語ではそれらの機能を提供する．

これらの XSLT による更新処理や Web サービスによ

る新規作成処理は１つ１つの単一の部分木に対して行う処理であり，これらの処理だけでは複数の部分木に対する処理は行えず柔軟性に乏しい．そこで複数の部分木に対する最も原始的な処理の１つとして，本言語では n 個の部分木を１つの部分木に再構成する結合処理についてもその機能を提供する．

これにより本言語の主要機能として，以下の３つの XML 形式のタグを規定する．

・<xslt-filter>タグ
・<wsdl-filter>タグ
・<concatenate>タグ

これらのタグは XML の規格に準じている．したがってこれらのタグを用いて処理を記述する場合，XML の規則に則った記述がなされなければならない（例えば開始タグで始まり終了タグで終わらなければならない，など）．またこれらのタグを含んだ文書は XML 文書でなければならない．

2.3.1. <xslt-filter>タグ

```
<!ELEMENT xslt-filter ANY>
<!ATTLIST xslt-filter
   xslt   CDATA   #REQUIRED
   eval   (eager | lazy )   “eager”
>
```

図 4 xslt-filter の DTD

xslt-filter のための DTD を図 4に示す．ここで，xslt-filter は高々１つ任意の子要素をもつ． このタグの開始タグと終了タグで囲まれている XML 文書に xslt 属性で指定された URI の XSLT スクリプトを適用する．その際，eval 属性に”eager”（先行評価）が指定されている場合は開始タグと終了タグで囲まれている XML 文書を先に一旦評価してから XSLT を適用する．eval 属性が”lazy”(遅延評価)の場合は評価せずに XSLT スクリプトを適用する．

2.3.2. <wsdl-filter>タグ

```
<!ELEMENT wsdl-filter ANY>
<!ATTLIST wsdl-filter
   wsdl   CDATA   #REQUIRED
   eval   (eager | lazy )   “eager”
>
```

図 5 wsdl-filter の DTD

wsdl-filter のための DTD を図 5に示す．ここで，wsdl-filre は高々１つの任意の子要素を持つ．このタグの開始タグと終了タグで囲まれている XML 文書を wsdl 属性で指定された URI の WSDL が示す Web サービスにパラメータとして引き渡す．その際，eval 属性に”eager”（先行評価）が指定されている場合は開始タグと終了タグで囲まれている XML 文書を先に一旦評価してから Web サービスに引き渡す．eval 属性が”lazy”（遅延評価）の場合は評価せずに Web サービスに引き渡す．

なお，本言語処理系も Web サービスとするため，このタグにより，他の Web サーバ上に配置されている本言語処理系のための WSDL を指定することで，本フレームワークから本フレームワークを利用することが可能になる．

2.3.3. <concatenate>タグ

```
<!ELEMENT concatenate ANY>
<!ATTLIST concatenate
   item   CDATA   #REQUIRED
   eval   (eager | lazy )   “eager”
>
```

図 6 concatenate のＤＴＤ

concatenate のための DTD を図 6に示す．ここで，concatenate は任意数の任意の子要素を持つ．このタグの開始タグと終了タグで囲まれている XML 文書の中の，item 属性で指定されたタグで囲まれ部分を抜き出して結合し，１つの XML 文書とする．このタグと item 属性で指定されたタグとの間にあるタグは無視される．またこのタグの子孫要素で，item 属性で指定されたタグを持たない要素も無視される．結合の際は item 属性で指定されたタグの

子要素については，その構成など一切問わない．

eval 属性に"eager"（先行評価）が指定されている場合は開始タグと終了タグで囲まれているXML文書を先に一旦評価してから抜粋，結合処理を行う．eval 属性が"lazy"（遅延評価）の場合は評価せずに抜粋，結合処理を行う．

3. 本言語による記述と動作例

ここでは設計した言語（タグ）の記述例とその動作について述べる．

3.1. <xslt-filter>タグと<wsdl-filter>タグ

図 7は<xslt-filter>タグ，および<wsdl-filter>タグを使った処理記述の例とその処理例を示している．図の最も上の四角内部にあるＸＭＬ文書が処理手順を記述した例であり，最も下の四角内部にあるＸＭＬ文書が処理結果であり，真中の２つの四角内部にあるＸＭＬ文書は処理の途中結果をＸＭＬ文書とした場合の例を示している．ここで，http://musium.ac.jp/search.wsdl には，博物館所蔵の標本情報を提供する Web サービスのインタフェース定義があり，http://gis-center.ac.jp/convert.wsdl は日本の地名（都道府県名，市町村名など）をその代表点の緯度経度に変換する Web サービスのインタフェース定義があり，http://www.foo.ac.jp/selection.xslt には，<location>タグとそれ以下の要素だけを抜き出す XSLT スクリプトがあるとする．

この例では「イリオモテヤマネコ」の標本が採取された位置の緯緯度を求める処理記述例を表している．ここでは，検索対象の博物館が提供する情報には，穂標本が採取された住所情報しかないために，住所から緯経度に変換するサービスを組み合わせて利用者が求める情報を得ている．

この例ではすべての<xslt-filter>タグ,<wsdl-filter>タグにおいて，eval 属性に"eager"（先行評価）が指定されている．したがって，一番内側の<wsdl-filter>タグから処理が実行され,博物館に問合せを行う.そして，必要な部分だけを切り出し，その結果を変換サービスに転送し，利用者が望む形に変換している．

この例のように，Web 上に公開されている XSLT スクリプトや公開されている Web サービスのインタフェース定義から必要なものを組み合わせて使うことが可能としている．

```
<root>
 <wsdl-filter
    wsdl="http://gis-center.ac.jp/convert.wsdll"
    eval="eager">
    <xslt-filter wsdl=
     "http://www.foo.ac.jp/selection.xslt"
     eval="eager">
     <wsdl-filter wsdl=
       "http://musium.ac.jp/search.wsdl"
       eval="eager">
       <name>
        Prionailurus bengalensis iriomotensis
       </name>
     </wsdl-filter>
    </xslt-filter>
 </wsdl-filter>
</root>
```

```
<root>
 <wsdl-filter
    wsdl="http://gis-center.ac.jp/convert.wsdl"
    eval="eager">
    <xslt-filter wsdl=
     "http://www.foo.ac.jp/selection.xslt"
     eval="eager">
     <recored>
       <common_name>
       Iriomote Cat</name>
       </common_name>
       <jp:location>Iriomote Island, Japan
       </jp:location>
     </recordt>
  </xslt-filter>
 </wsdl-filter>
</root>
```

```
<root>
 <wsdl-filter
    wsdl="http://gis-center.ac.jp/convert.wsdl"
    eval="eager">
      <jp:location>Iriomote Island, Japan
      </jp:location>
 </wsdl-filter>
</root>
```

```
<root>
  <location>
    <LAT>24</LAT>
    <LNG>123</LNG>
  </location>
</root>
```

図 7 処理記述例とその実行例その１

```
<root>
  <wsdl-filter
    wsdl="http://othre.ac.jp/processer.wsdl"
    eval="eager">
    <xslt-filter wsdl=
     "http://portal.ac.jp/list.xslt"
     eval="eager">
       <name>
        Prionailurus bengalensis iriomotensis
       </name>
    </xslt-filter>
  </wsdl-filter>
</root>
```

```
<root>
 <wsdl-filter
    wsdl="http://othre.ac.jp/processor.wsdl"
    eval="eager">
   <wsdl-filter wsdl=
     "http://musium-a.ac.jp/search.wsdl"
     eval="eager">
     <name>
        Prionailurus bengalensis iriomotensis
     </name>
    </wsdl-filter>
   <wsdl-filter wsdl=
     "http://musium-b.ac.jp/search.wsdl"
     eval="eager">
     <name>
        Prionailurus bengalensis iriomotensis
     </name>
    </wsdl-filter>
 </wsdl-filter>
</root>
```

```
<root>
 <recored>
 <common_name>IriomoteCat</common_name>
 <jp:location>Iriomote Island, Japan
 </jp:location>
 </recordt>
 <recored>
  <location>
    <LAT>24</LAT>
    <LNG>123</LNG>
  </location>
 </recordt>
</root>
```

図 8 処理記述例とその実行例その２

図 8 に別の例を示す．ここで，http://other.ac.jp/processor.wsdl にはこの言語処理系のサービスのためのインタフェース定義があり，http://portal.ac.jp/list.xsl には与えられたきーワードから複数の博物館に対して標本情報を問い合わせるような処理記述を生成する XSLT スクリプトがあるとする．この例は，１つの問合せから複数のデータソースを対象とした検索を実現するサービスが記述できる例である．また，この例では，本フレームワークから再帰的に呼び出している．このように，本フレームワークから自分自身を呼び出すことことにより，ある関数の複数の引数の評価を各々別の処理プロセッサに割り当てることにより，評価の並列処理が可能となる．

3.2. <concatenate>タグ

```
<root>
  <concatenate item="location" eval="quote">
    <IriomoteCat>
      <name>Iriomote Cat</name>
      <location>
        <LAT>24 23'1"</LAT>
        <LNG>123 44'9"</LNG>
      </location>
    </IriomoteCat>
    <data>
      <common-name>Iriomote Cat
      </common-name>
      <name>
      Prionailurus bengalensis iriomotensis
      </name>
      <other>
        <location>
          <LAT>24</LAT>
          <LNG>123</LNG>
        </location>
      </other>
    </data>
  </concatenate>
</root>
```

```
<root>
    <location>
      <LAT>24 23'1"</LAT>
      <LNG>123 44'9"</LNG>
    </location>
    <location>
      <LAT>24</LAT>
      <LNG>123</LNG>
    </location>
</root>
```

図 9 処理記述例とその実行例その３

<concatenate>タグと</concatenate>タグで囲まれた XML 文書の中から「location」という名前のタグを抜き出して結合する，という例を図 9に示す．

<concatenate>の子である<IriomoteCat>には item 属性で指定された<location>タグがあるので<location>タグと<location>タグで囲まれた部分を抜き出す．<IriomoteCat>には他にも<name>タグがあるが item 属性で指定されたタグ名とは異なるので無視する．また<IriomoteCat>タグ自身も無視する．<concatenate>の別の子である<data>には，<data>の子の<other>のさらに子要素として item 属性で指定された<location>タグがあるので<location>タグと</location>タグで囲まれた部分を抜き出す．<data>の他の子要素，および<data>, <other>自身は無視する．

その結果，<Iriomote Cat>の子である<location>と<other>の子である<location>が結合された XML 文書が生成される．

この例は，別途 XSLT スクリプトを記述し，xslt-filter によりそのスクリプトを提供することにより同様の実現ができる．もし，結合をしたい部分木が本フレームワークにより動的に変化する場合には，それらが確定しない限り XSLT スクリプトが適用できない．このフィルタでは，指定された要素以下の要素を結合することができるため，複数の部分木の評価を同時に行うといった並列処理を実現することが可能となる．

4. 分散 XML 処理言語の処理系の実装

ここでは設計した分散 XML 処理言語を解析，処理する処理系について述べる．

言語の解析，処理を行う処理系については Xi[6]の機能拡張として実装する．

Xi（eXtend it）は非営利団体である横浜ベイキットによって開発された，XML 文書を生成するための XML 形式のプログラミング言語である．Xi は仕様が公開されており，また Xi の処理系である Xi インタプリタについても「BXi（Baykit Xi Processor）」というオープンソースである処理系が公開されている．

Xi は XML 形式の言語であり，「Xi 関数」と呼ばれる特別な要素の組み合わせによりプログラムが構成される．Xi に標準に用意されている Xi 関数は開始タグが<xi:～>の形で，終了タグが</xi:～>の形で記述される．これらの Xi 関数が Xi インタプリタによって解析，処理される．Xi には標準の関数として 19 の関数が用意されている．

また，Xi には標準で用意されている機能以外にも，タグライブラリや Ninja といったいくつかの拡張モジュールを備えている．これらの拡張モジュールを Xi Extension と呼ぶ．この内タグライブラリは Xi プログラムで利用できるタグ（Xi 関数）を拡張し，Ninja は Xi プログラムで利用できるオブジェクトを拡張するものである．

我々は，本言語系で定義した関数を Xi の機能拡張方法の１つである「タグライブラリ」型の機能拡張，すなわち，<xslt-filter>，<wsdl-filter>，<concatenate>の各タグに対する処理のみを実装する形により，実現した．

Xi のタグライブラリの機能拡張には Java 言語を用いる．これは Xi そのものが Java によって実装されているためである．また Java による実装の副次的な効果として， Java の動作環境さえあればその実行環境を選ばない，という特徴を持つ．

ここでは，<wsdl-filter>タグの実装について詳しく述べる．以下に<wsdl-filter>タグの主要処理である WSDL を利用した Web サービスの呼び出し処理を，一部擬似コードを用いて示す．ただし，TreeFragment 型変数 result は <wsdl-filter> タグの処理結果が，WSDLNinjaFactory 型変数 wsdlFactory は Xi の機能拡張「WSDL-Ninja」に含まれる WSDL 用の WSDLNinja オブジェクトが WSDLNinja 型変数 wsdlNinja には，Xi の機能拡張「WSDL-Ninja」に含まれる WSDL に関する処理オブジェクトがそれぞれ格納されるものとする．

```
wsdlFactory = new WSDLNinjaFactory();
wsdlFactory.initializeNinja(this.engine);
wsdlNinja = wsdlFactory.newObject(
    利用する WSDL の URL,
    利用するサービス名 ); ……①
Ninja[] wsdlParameter =
    new Ninja[<wsdl-filter>タグの子要素の数 ];
for( int i = 0; i < <wsdl-filter>の子要素の数; i++ ) {
  Node chlid;
  if( この関数に eager が指定されている){
    child = <wsdl-filter>タグの i 番目の子要素;
  }else{
    child =
    eval(<wsdl-filter>タグの i 番目の子要);
    // 評価した結果を child に格納
  }
  wsdlParameter[i] = child.toString();……②
}
results = convert(wsdlNinja.doInvoke(
利用する処理名, wsdlParameter));……③
```

この処理では WSDLNinja オブジェクトの生成時に利用する WSDL の URI，および利用する Web サービスの名前を指定し（①），<wsdl-filter>タグの子要素を Web サービスへのパラメータとして準備し（②），その後 Web サービスへのアクセスを行い，タグの処理結果となる型に変換している（③）．

これらの実装を以下の環境上に行って動作を確認した．その結果目的のする XML 文書が生成された．

- CPU
  Intel® Celeron 466MHz
- Memory
  256MB
- Operating System
  Microsoft® Windows2000 Professional

- Java Environment
  Java2™ SDK(Software Development Kit),
  Standard Edition version 1.3.1
  Java version 1.4.1_01
- Xi Environment
  Baykit Xi Processor version 1.1.4

5. おわりに

本論文では Web 上に分散する XML 配信サービスを結合したり，またはそれらの処理に付加価値を与えたりして新しいサービスの提供を容易にするメタフレームワークとしての XML 形式の言語，およびその処理系について述べた．また，設計に基づいて実装した処理系，およびその処理系の評価についても述べた．処理系の実装を Xi の機能拡張としての実装とすることで開発効率が向上し,開発に必要な工数を短縮できた．評価としては設計，開発した処理系を用いて実施にいくつかの XSLT の適用や Web サービスへのアクセス，XML 要素の結合処理などを行い,それらが設計で要求された機能を満たしていることを確認し，またその有効性の確認も行った．特に，本研究で設計した XML 形式の言語はそれぞれのタグを単体で用いるより，それらのタグを組み合わせて用いた方がより有用な効果をもたらすことも確認できた．

これらのフレームワークを利用することで今後需要が予想される複数の XML 配信サービスの結合や，それらのサービスへの付加価値の付与などに，統一的な手法で容易に対応することができることが期待される．

本フレームワークはまだ発展途上の段階であり，向上の余地は多分にあると考えられる．現在，今後の課題として以下のものを考えている．

・複数の部分木に対する処理機能の拡充

現時点では複数の部分木に対する処理としては部分木の結合処理（<concatenate>タグ）しか実装されておらず，機能として不十分である．結合処理以外にも例えば複数の部分木を同時に並列実行させる機能など，柔軟な処理を行うために必要な機能の実装が必要であると考える．

・SAX ベースの処理系への移行

現在の実装は Xi を基盤とした DOM ベースの実装になっている．DOM の実装と SAX の実装では処理速度の面などにおいて SAX の実装の方が優位である．SAX にも欠点はあるが本フレームワークでは SAX の短所はほとんど影響しない．他に，本フレームワーク上では，ある関数の引数をそれぞれ別の処理系を用いて並列に評価することが可能であるが, そのためには XML 文書のパースを一旦終了しなければ処理ができない DOM では処理の並列化を行えない．これらの理由により, 処理系を SAX ベースの実装へ移行するべきであると考える．

・生物情報への適用

現在，生物多様性情報に関する情報交換が盛んに研究されている[12]．今後インターネットに接続された博物館や研究機関などが，生物種に関するデータベースを公開するようになる．このとき,必要な情報を様々なデータベースに問合せ，それらを利用者が望む形式（例えば表形式や地図形式など）変換する必要性がある．データベースとのデータ交換方式には XML が使われると思われる．このとき，スキーマ変換のための XSLT スクリプト，複数のデータベースへの問合せに変換するスクリプト，検索結果を表形式や地図を表す SVG[4]に変換するスクリプトなどを提供することで，柔軟なデータ処理を可能とする環境を提供するために，本フレームワークを適用していくことは今後の課題である．

文　献

[1] T.Bray, J.Paoli, C.M.Sperberg-McQueen and E.Maler, Extensible Markup Language (XML) (Second Edition). W3C Note, http://www.w3.org/TR/REC-xml,Oct.,2000.

[2] E.Christense, F.Curbera, G.Meredith and S.Weerawarana, Web Services Description Language (WSDL) 1.1. W3C Note, http://www.w3.org/TR/wsdl, Mar.,2000.

[3] J.Clark, XSL Transformation (XSLT) Version 1.0. W3C Note, http://www.w3.org/TR/xslt, Nov., 1999.

[4] J.Ferraiolo, J. Fujisawa and D. Jackson. Scalable Vector Graphics (SVG) 1.1 Specification. W3C Note, http://www.w3.org/TR/SVG11, Jan., 2003.

[5] Java™ 2 Platform, Standard Edition (J2SE™), http://java.sun.com/j2se/

[6] 横浜 Baykit, http://www.baykit.org/

[7] XML Press Vol.6, 技術評論社

[8] Software Design 2002.10, 技術評論社

[9] David Hunter, Kurt Cagle, Dave Gibbons, Nikola Ozu, Jon Pinnock, Paul Spencer 著, 風工舎 訳: エキスパートから学ぶ XML 実線プログラミング, インプレス

[10] 林晴比古 著: 新 Java 言語入門 ビギナー編, ソフトバンク パブリッシング

[11] 林晴比古 著: 新 Java 言語入門 シニア編, ソフトバンク パブリッシング

[12] 佐藤 聡, 伊藤 希, 小野 哲, 柁原 宏, 志村 純子 著: 生物種情報データベースのためのデータ交換方式, 論文(情報処理学会研究報告 DBS128-34, 2002)