XML プログラミング言語 Xi(ザイ)の完全な仕様策定とその実装

- 美しいツリー言語の提案 -

1.背景

Web アプリケーションの構築に際し、プログラマが選択できるプログラミング言語には様々な種類がある。例えば、C や C++、Java などの基本的なプログラミング言語、Perl や Python、Tcl、最近では Ruby などのスクリプト言語、PHP や ASP のような HTML 埋め込み型テンプレート言語などがある。また、後発の HTML 埋め込み型の簡易性を基本的なプログラミング言語で利用できる JSP のような方法も生まれた。

これらの言語に共通する欠点は、プログラミング言語と HTML が混在し、可読性が著しく低いことである。そこで、横浜ベイキット(http://www.baykit.org)では、このような問題を解決するための新しいツリー言語「Xi(ザイ)」を公開している。Xi はプログラムを XML 形式で記述することで言語の混在問題をきれいに解消している。

ところが、Xi にも問題点がないわけではない。Xi の問題点は、言語仕様が完全でないことである。Xi のタグは「関数」とは呼ばれているが、一般的なプログラム言語の関数とは程遠い。そもそも、ユーザーが関数を定義することもできないし、関数の引数の評価方法も明確に定められていない。このように、Xi はプログラミング言語としては不十分な言語であった。

2.目的

本プロジェクトの目的は、Xi の完全な言語仕様を策定することと、その処理系を実装することである。これにより、名実ともに新しいツリー処理言語の誕生が期待できる。

3.開発の内容

Xi を完全な言語仕様にするためにポイントとなるのは関数の定義方法を明確にすることである。また、現状の Xi がサポートしている XPATH の構文は言語仕様を複雑にする原因となるので、言語仕様とは別の枠組みでサポートすることにした。

実装については、Xi コードを Scheme のコードに変換し、変換されたコードを Scheme インタプリタ上で実行し、結果を XML 形式にレンダリングする方法を試みた。ただし、ツリー構造とリスト構造にはデータ構造上に大きな違いが

あるため、変換過程で「Fish」と呼ばれる、特殊なツリー構造をした構文木を作成することにした。

3-1.仕様について

仕様を策定するに当たって、いろいろな問題点が浮上してきた。それは、言語仕様をシンプルにするとタグを多く用いなければならず、とても手で記述できるようなプログラムにならないということである。ちょうど Lisp のプログラムを記述するのに、括弧を使わずわざわざ XML のタグを使用するようなものである。また、XPATH を使えないということは、四則演算やノードの参照などの記述が冗長になることは間違いない。

そこで、言語仕様はできるだけシンプルではあるが、構文の追加を行えることにした。このシンプルな言語仕様を「基本構文」と名づけ、より扱いやすい構文を「拡張構文」と名づけた。拡張構文はより記述のしやすい構文で、冗長なタグ表記を排除し、XPATH のサポートも行える。ただし、言語仕様に一貫性を持たせるためには、拡張構文は基本構文に変換できなければならないので、変換方法も定義した。

例えば、基本構文によって定義される関数は次のようになる。

```
<cr:var>
 <cr:name>fact</cr:name>
 <cr:lambda>
   <cr:args>
     <cr:arg>x</cr:arg>
   </cr:args>
   <cr:if>
     <eq><cr:name>x</cr:name><cr:number>0</cr:number></eq>
     <cr:number>1</cr:number>
     <mul>
       <cr:name>x</cr:name>
       <sub><cr:name>x</cr:name><cr:number>1</cr:number></sub>
     </mul>
    </cr:if>
  </cr:lambda>
<cr:var>
```

リスト 1. 基本構文での関数定義

これを拡張構文で記述すると、次のようになる。

```
<xi:function name="fact" args="x">
   <xi:if test="$x = 0">
      <xi:then>
           <xi:value-of select="1"/>
      </xi:then>
      <xi:else>
           <xi:value-of select="$x * fact($x - 1)"/>
      </xi:else>
    </xi:if>
</xi:function>
```

リスト 2. 拡張構文での関数定義

Xi の構文を定義した仕様は、
http://www.baykit.org/~makotos/cgi-bin/fish.cgi?Spec%3aXi&l=jp
で公開した。

データモデルについては、「Fish」と呼ばれる特殊なツリー構造を考え、それを採用した。このツリーは、二分木を拡張したものである。ツリーを構成する各構造を**ノード**と言う。ノードは**枝**と**葉**に分かれる。枝は、名前をもち、子ノードおよび隣ノードへの参照を持つ。葉は任意のデータを持ち、隣ノードへの参照を持つ。

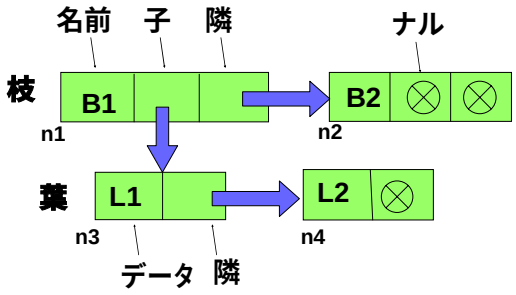

図 1. Fish データ構造

図 1 を例にとると、n1, n2 は枝の例であり、それぞれ名前「B1」、「B2」を持つ。n1 の子は n3 であり、これは葉である。n3 はデータ「L1」を持ち、n3

の隣は n4 である。n2 は子および隣は持たない。また n4 は隣を持たない。このような場合、参照は**ナル**であるという。

Fish のデータモデルについての仕様は、

http://www.baykit.org/~makotos/cgi-bin/fish.cgi?Spec%3aFish&l=jp

で公開した。

3-2.実装について

実装はコンパイラ方式で行った。コンパイラの実装にはすべて Scheme を用いた。Xi コードのコンパイル、および実行の流れは図 2 のようになる。

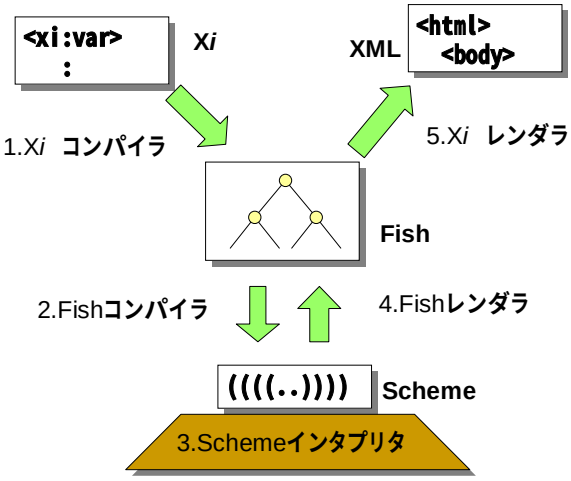

図 2. Xi コンパイラの構成

1.まず、Xi コンパイラによって Xi コードを Fish 形式に変換する
2.Fish 形式を Fish コンパイラによって実行可能な Scheme コードに変換する
3.変換された Scheme コードを Scheme インタプリタ上で実行する
4.実行結果を Fish レンダラによって Fish 形式に変換する
5.Fish 形式の実行結果を Xi レンダラによって Xi 形式(XML 形式)に変換する。

つまり、コンパイラのアーキテクチャとしては、Xi のソースを Fish とよばれる解析木に変換し、その後 Scheme コードとよばれるアセンブリ言語に変換する、と思えばよい。実装の詳細は

http://www.baykit.org/~makotos/cgi-bin/fish.cgi?Spec%3aImplementation&l=jp

で公開した。また、作成されたコンパイラは

http://www.baykit.org/~makotos/xi.tar.gz

で公開した

4.従来の技術との相違

従来のプログラミング言語との相違は、言語の混在問題が解消されていることである。このため、HTML 文書を生成する Web アプリケーションの世界では可読性の優れたコードを書くことができる。

また、従来の Xi との相違という点ではデータモデルが確立されたこと、プログラミング言語の意味が与えられたことである。これにより言語仕様が明確になり、一貫性のある動作が保証される。

5.期待される効果

今回のプロジェクトにより、従来 Xi の持っていた言語仕様の汚さが解消され、よりわかりやすい言語になったと期待される。また、学術的に見た言語の美しさの追求と、ユーザーにとっての扱いやすさの両方を満たすことができたので、普及も加速すると期待できる。

6.普及の見通し

Xi は横浜ベイキットによりオープンソースで公開されている。また、現在（株）アルゴクラフトでは、Xi を使用したシステム構築を手がけており、事例を増やしていく予定である。また、商用化も視野に入れており、普及を加速させたい。

7.開発者名

川道亮治（横浜ベイキット　kawao.mito-pub04@submit-asap.org）
佐藤誠（横浜ベイキット　makotosato2@yahoo.co.jp）