Webアプリケーションの汎用化のための中間表現の提案と実装

早川　智一†‡　　長谷川　慎哉‡　　疋田　輝雄‡

（株）ティージー情報ネットワーク†　　明治大学理工学研究科‡

1. はじめに

汎用性・移植性の高いWebアプリケーションを作成するために必要な，XMLをベースとする最小限の要素および記述方式（以下，中間表現）を定義し提案する．さらに既存のWebアプリケーションからの移植を考慮し，Webアプリケーションから中間表現（およびその逆方向）へ自動翻訳するための処理系をJavaで実装した．

Webアプリケーションはその構成技術（XHTML, CSS, JavaScript）による環境非依存性から，一度作成してしまえばブラウザが存在する任意の環境で動作すると考えられがちであるが，Webアプリケーションのユーザインターフェースが高度化するにつれ，ブラウザごとにその解釈・挙動の違いが顕在化している．特に業務としてWebアプリケーションを採用・開発している企業にとっては，アプリケーション自体が資産であるため，システムの拡充改善と，汎用性を高く保つことが望まれている．

今回提案する中間表現は主に，(1) 開発容易性，(2) 他言語への移植容易性，(3) 他言語からの移植容易性を実現することを目標としている（図1.1）．

2. 関連技術

現在一般的に利用されている手法としては，jQuery等に代表されるクロスブラウザを実現するためのフレームワークを使用し，ブラウザ間の差異を吸収することが行なわれている．しかしこれらのフレームワークを利用した場合，今度はフレームワークに対する依存性が発生してしまい，移植性は高まるが汎用性を高めたとは言いにくい．

XMLをベースとしたユーザインターフェース情報の定義および，Webアプリケーションの実現には様々な方式が考案されており，前者としてはMicrosoft社のXAML [5]やMozilla FoundationのXUL [6]が，後者としてはAdobe社のMXML [1]やLaszloSystems社のLZX [3]がある．これらに共通する特徴としては，高い表現能力により多機能なアプリケーションを記述することができるが，その反面，要素技術の連携範囲や移植性について難しい点がある．

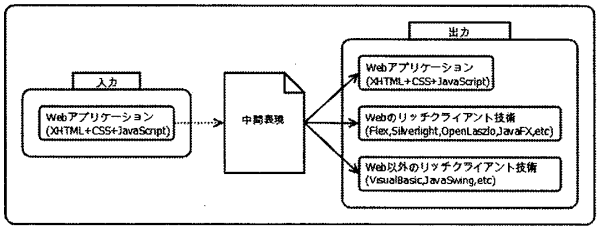

図1.1　変換のための中間表現

3. 中間表現の概要

```
<ui>
    <layout>
        <!--レイアウト情報-->
    </layout>
    <style>
        <!--スタイル情報-->
    </style>
    <behavior>
        <!--振る舞い情報-->
    </behavior>
</ui>
```

図3.1　中間表現の基本構造

中間表現はXMLをベースとした記述方式で表現される（図3.1）．表3.1はその要素を示す．

表3.1　中間表現の代表的な要素

要素名	親要素	意味
title	layout	アプリケーションのタイトル
heading	layout	見出し
block	layout	汎用ブロック要素
text	layout	テキスト文字列
button	layout	ボタン
image	layout	画像
font	style	フォントの指定
color	style	色の指定
onclick	behavior	クリック時に発生

Design and Implementation of Intermediate Representation for Generalization of Web Applications.
† TG Information Network Co., Ltd.
‡ School of Science and Technology, Meiji University.

中間表現のルートタグは<ui>である．この内部にレイアウトを定義するための<layout>，スタイルのための<style>，振る舞いのための<behavior>を含む．これらはそれぞれ，Web アプリケーションにおける XHTML，CSS，JavaScript に対応する．

図 3.2，3.3，3.4 は一連の変換を示している．図 3.5 は図 3.2 に対応する画面で，入力においてボタンをクリックした際のキャプチャである．画面 3.6 は変換後の Laszlo LZX においてボタンをクリックした際のキャプチャで，Flash である．

```
<html>
  <head><title>サンプル</title></head>
  <body>
    <img src="duke.jpg"/>
    <p style="font-style:italic">画像とボタンのサンプル
</p>
    <button onclick="alert('ボタンが押されました');">ボ
タン</button>
  </body>
</html>
```

図 3.2　XHTML によるファイル

```
<ui>
  <layout>
    <title>サンプル</title>
    <image src="duke.jpg"/>
    <block id="id1"><text>画像とボタンのサンプル
</text></block>
    <button id="id2"><text>ボタン</text></button>
  </layout>
  <style>
    <font style="italic" target_id="id1"/>
  </style>
  <behavior>
    <function event="onclick" target_id="id2">
      alert('ボタンが押されました');
    </function>
  </behavior>
</ui>
```

図 3.3　中間表現

```
<canvas title="サンプル">
  <simplelayout axis="y" />
  <alert name="a1">ボタンが押されました</alert>
  <view resource="duke.jpg"/>
  <view fontstyle="italic">
    <text>画像とボタンのサンプル</text>
  </view>
  <button onclick="a1.open()">ボタン</button>
</canvas>
```

図 3.4　出力された LZX ファイル

4. 処理系の実装

処理系への入力として与えられる XHTML/CSS/JavaScript は，それぞれ DOM，CSStoXML[4]，JavaCC[2]等を用いて実装された処理系を経由して，XML 文書として解釈できる一つの木構造として再構成される．この後，処理系内部で保持している要素・属性の置換表に基づいて，木の変形が行われ，これが中間表現となる．中間表現から出力への変換も処理系内部の置換表を用いて行われる．

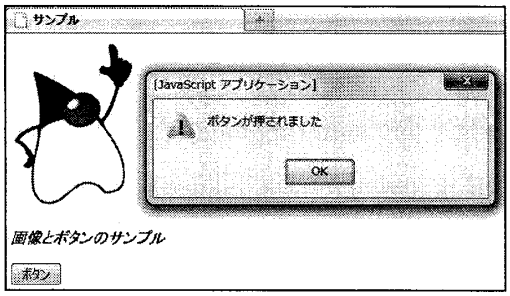

図 3.5　XHTML の画面

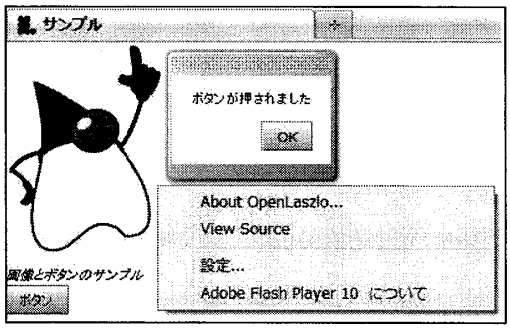

図 3.6　LZX の画面

5. おわりに

Web アプリケーションの一般化の方法論として XML に基づいた中間表現を提示した．ただし，現時点では UI 情報の中間表現としての位置付けが強く，動作に関する表現力の向上は今後の課題である．他言語への変換については言語仕様の相違等を吸収するための変換規則を拡張することで対応を図る．今後の継続研究を通じて，多くの Web アプリケーションが特定のベンダーや実装技術に縛られることなく，汎用的に実現されるようになることを目指したい．

参考文献

[1] Adobe, About MXML, http://livedocs.adobe.com/flex/3/html/help.html?content=mxml_2.html
[2] JavaCC, https://javacc.dev.java.net/
[3] Laszlo Systems, Inc., LZX Reference Manual, http://www.openlaszlo.org/lps/docs/reference/
[4] Jukka Mäntylä, CSStoXML v1.02, http://appro.mit.jyu.fi/tools/csstoxml/
[5] Microsoft, XAML Overview, http://msdn.microsoft.com/en-us/library/ms752059.aspx
[6] Mozilla Foundation, XUL (XML User Interface Language), https://developer.mozilla.org/En/XUL