M-048

Flex アプリケーションの iOS 環境での HTML5 変換による実行

吉 賀 祥 太[†1] 早 川 智 一[†1] 疋 田 輝 雄[†1]

Execution of Flex Applications under the iOS Platform by HTML5 Transformation

Shota Yoshika,[†1] Tomokazu Hayakawa[†1] and Teruo Hikita[†1]

1. はじめに

今日，iPhone・iPad などの多機能携帯端末（携帯端末）の登場により，RIA(Rich Internet Application) の必要性が大きく変わりつつある．すなわち，従来のコンピュータ上だけでなく，携帯端末上でも RIA が利用されるようになってきている．特に企業では，RIA として構築された業務アプリケーションを携帯端末上で利用することで，業務の効率化を図る動きがある．

一方で，企業で主に利用されている RIA 技術として Flex[2] がある．Flex は表現力・操作性に優れたフレームワークであり，業務アプリケーションの構築に広く用いられている．また，Flex は実行環境に Flash Player を採用することで，OS・Web ブラウザなどの環境の違いに依らない同一な動作を提供する．この Flex の特徴は大きなメリットとなるが，前述の iPhone・iPad が採用している OS（iOS）上では動作しないという問題がある．これは偏に，Flex が 1 つのベンダに依存しているために起こる問題である．

この問題を解決する手段としては，iOS 上で動作する他の RIA 技術への変換が考えられる．しかし，これは手作業が基本となるため容易ではなく，大規模なシステムの場合には大きなコスト負担となる．

我々は，Flex による業務アプリケーションを iOS 上で動作させる方法として，HTML5[5] への自動変換を提案する．ここで，変換の自動化は前述のコストの低減に寄与すると考える．何故ならば，HTML5 は近年注目されているオープン標準な RIA 技術であり，iOS 上でも動作する．また，HTML5 の使用は，ベンダ依存の問題を解決する．

本論文の構成は以下のとおりである．2 節では関連研究について述べる．3 節では変換対象となる Flex アプリケーションについて述べる．4 節では例を用いて変換の様子について述べる．5 節では変換システムの設計について述べる．6 節では変換システムの実装について述べる．7 節では変換システムの評価について述べる．8 節では考察と今後の展望について述べる．

2. 関連研究

Flash アプリケーションを iOS 環境で動作させる研究として，Adobe Labs の Wallaby[1] がある．Wallaby は，Flash Professional により作成された Flash アプリケーションを HTML 形式に変換することで動作を実現している．しかし，これは Flex アプリケーションを変換対象として含まない．

また，Adobe Systems は，Flex によるモバイルアプリケーション開発の研究[3] を行っている．これは，iOS も含めたモバイル環境用の新たな Flex コンポーネントを用いてアプリケーションを開発するためのものである．しかし，これは，新規開発が対象であり，既存のアプリケーションを iOS 用に自動変換するものではない．

早川らは RIA の汎用化の方法として，中間表現と変換用フレームワーク[7)–9)] を提唱している．我々の変換システムも開発の効率化のためにこのフレームワークを採用した．

3. 本システムの変換対象

Flex は，構成技術として MXML や ActionScript 言語を持つ高機能な RIA 技術である．また，Flash Player を実行環境とするため，Web ブラウザ上だけでは再現不可能な様々な機能を備えている．従って，これらをすべて HTML5 で再現することは難しく，変換対象にいくつかの制限を設ける必要がある．

そこで我々は Visual Basic[6] と同程度の Flex アプリケーションを変換対象とした．なぜならば，Visual Basic で構築された業務アプリケーションが，RIA 技術の普及とともに RIA として移植されるようになったためである．ここで Visual Basic のバージョンとしては，企業で広く使用されている 6 を想定した．

また，Flex は MXML による UI 情報と ActionScript 言語によってアプリケーションを構築しているが，本システムは現時点では UI の変換のみを対象としている．

4. 変 換 例

本節では，変換システムによる変換例を示す．ここで，HTML5 の実行環境である Web ブラウザとしては iPhone・iPad で標準搭載されている Safari を選択した．

図 1 が変換例である．左が変換前の Flex アプリケーションであり，右が変換後の HTML5 アプリケーションである．例として，認証ページを模したアプリケーションとした．レイアウトに若干の差異はあるものの，すべての要素の変換ができていることがわかる．

図 2 が変換前の MXML のソースコード，**図 3**・**図 4** がそれぞれ変換後の HTML・CSS のソースコードである．MXML の `Image` 要素から HTML5 の `img` 要素や，MXML の `Button` 要素から HTML5 の `button` 要素などのページを構成する個々の要素が変換されていることがソースコードからもわかる．特に MXML には `VBox`・`HBox` などの要素があり，これらは中に含

†1 明治大学理工学研究科
School of Science and Technology, Meiji University.

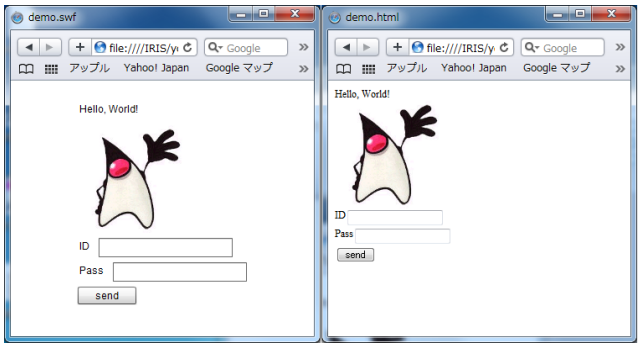

図 1　Flex から HTML5 への変換例
Fig. 1　Example of transformation from Flex to HTML5.

```
<?xml version="1.0"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml">
    <mx:VBox>
        <mx:Text>
            <mx:htmlText>
                <![CDATA[<p>Hello,World!</p>]]>
            </mx:htmlText>
        </mx:Text>
        <mx:Image source="@Embed(source='duke.png')"/>
        <mx:HBox>
            <mx:Label text="ID"/>
            <mx:TextInput/>
        </mx:HBox>
        <mx:HBox>
            <mx:Label text="Pass"/>
            <mx:TextInput/>
        </mx:HBox>
        <mx:Button label="send"/>
    </mx:VBox>
</mx:Application>
```

図 2　変換前の MXML ソースコード
Fig. 2　MXML source code before transformation.

む要素を垂直・水平に表示する機能を持つ．この例ではページ全体の要素を VBox を用いて垂直方向に配置し，その中に HBox が 2 つ存在している．HBox 内では要素を水平方向に配置するため，図 1 の ID・Pass のラベルから始まる列が水平方向に表示されていることがわかる．変換結果である図 1 右でも，それを反映した表示となっている．これらの配置は図 4 の CSS 内で .Vertical，.Horizontal（クラスセレクタ）で表される記述によって再現されている．

5. 設　　計

5.1 変換システムの概要

本システムは Java 言語により構築されており，Flex アプリケーションの MXML ファイルを入力として，HTML ファイルと CSS ファイルを出力する（図 5）．

また，変換の手順としては以下のような過程を経る．

(1)　MXML の入力
(2)　MXML の解析
(3)　要素変換
(4)　HTML の構築
(5)　HTML の解析
(6)　CSS の構築

```
<html>
    <head>
        <META http-equiv="Content-Type" content="text/html; charset=UTF-8">
        <style type="text/css">
            <!--@import url('container.css');-->
        </style>
    </head>
    <body>
        <div class="MxApplication Vertical">
            <div class="VBox Vertical">
                <div><p>Hello, World!</p></div>
                <img src="duke.png">
                <div class="HBox Horizontal">
                    <span>ID</span>
                    <div><input type="text"></div>
                </div>
                <div class="HBox Horizontal">
                    <span>Pass</span>
                    <div><input type="text"></div>
                </div>
                <div><button>send</button></div>
            </div>
            </div>
    </body>
</html>
```

図 3　変換後の HTML ソースコード
Fig. 3　HTML source code after transformation.

```
.Vertical {
    display:box;
    display:-webkit-box;
    display:-o-box;
    display:-moz-box;
    box-orient:vertical;
    -webkit-box-orient:vertical;
    -o-box-orient:vertical;
    -moz-box-orient:vertical;
}

.Horizontal {
    display:box;
    display:-webkit-box;
    display:-o-box;
    display:-moz-box;
    box-orient:horizontal;
    -webkit-box-orient:horizontal;
    -o-box-orient:horizontal;
    -moz-box-orient:horizontal;
}
```

図 4　変換後の CSS ソースコード
Fig. 4　CSS source code after transformation.

(7)　HTML・CSS の出力

図 6 に本システムの概要クラス図を示す．2 節で述べたとおり本システムは早川らのフレームワークを用いており，それに準じたクラス構成となっている．以下にそれぞれのクラスの説明を示す．

DocumentApplication
　MXML・HTML 文書を抽象化したもの
DocumentApplicationIOImpl
　文書の入出力を行う
DocumentTranslator
　文書の変換を行う
DocumentVisitor
　文書の解析を行う
NodeTransformer
　文書内の構成要素の変換を行う

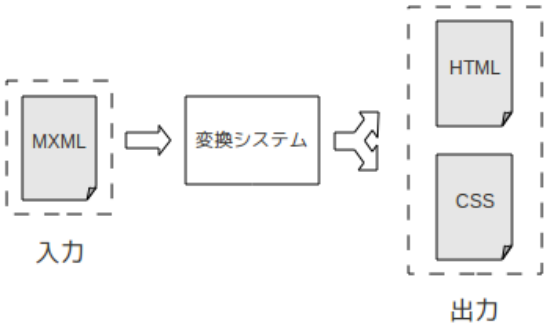

図 5　変換の概要
Fig. 5　Overview of the system.

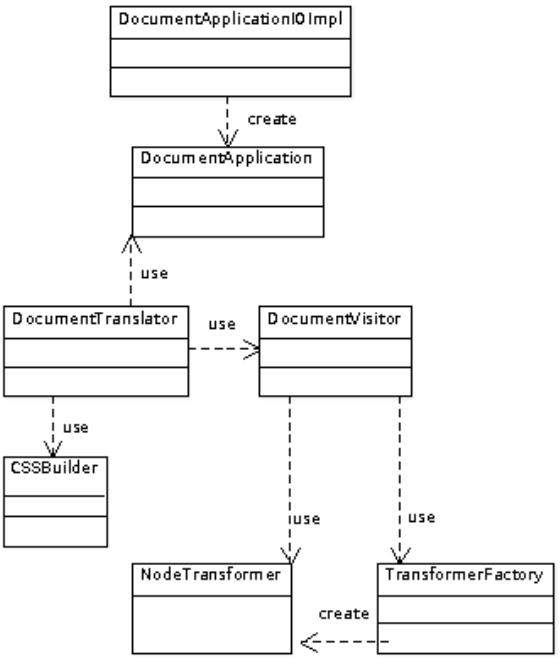

図 6　本システムの概要クラス図
Fig. 6　Class diagram of the system.

CSSBuilder
　CSS ファイルを構築する

5.2　要素の変換方法

5.1 の手順の (3) では MXML 文書内の要素を HTML5 の要素に変換することで文書の変換を実現している．ここで，MXML の要素には主に次の 2 種類が存在する．

(1)　コントロール　　ボタンやテキストなどの部品
(2)　コンテナ　　他の要素を包含し，レイアウトを施す

ここで，コントロールの場合は，HTML5 の対応する要素とほぼ 1 対 1 で変換することができるが，コンテナの場合，以下の理由のために HTML5 の要素だけでは変換不可能である．

図 7 は，`HBox` を HTML5 の形式に変換する方法を表している．`HBox` は，他の要素をグループ化して包含し，それらを水平方向に配置する機能を持つコンテナである．図からも分かるようにこのようなレイアウト機能を持つ MXML 要素は HTML5 要素よりも大きな機能を持つことになる．そこで，本システムでは，そのレイアウト機能を CSS を用いて再現した．他の要素を包含しグループ化する機能を持つ `div` 要素と，レイアウト機能を持つ CSS により，1 つの MXML 要素を対応させている．

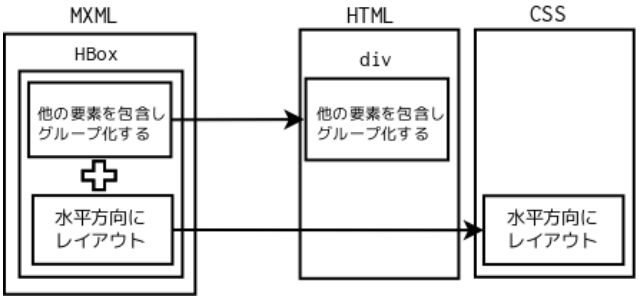

図 7　MXML の HBox の HTML5 への変換
Fig. 7　Transformation of MXML HBox to HTML5.

5.3　スタイルの変換方法

5.2 で述べたとおり本システムでは，レイアウト機能を再現するために CSS を使用するが，要素変換の段階では後の CSS 構築のために，変換するレイアウト情報を HTML 文書の中に記憶する必要がある．

本システムでは，記憶場所として `class` 属性を使用する．`class` 属性には対応するコンテナ名とレイアウト名を記述し，これを基に CSS の適用を行う．ここでコンテナ名とレイアウト名を分けた理由は，一部のコンテナは独自のレイアウト情報を持つことができるためである．

また，CSS の構築には，まず HTML 文書を解析し `class` 属性に記憶されたスタイル情報を抽出する必要がある．その後，抽出されたスタイル情報を基に，CSS プロパティ（CSS 形式のスタイル情報）を記述し，クラスセレクタにより該当の HTML5 要素にスタイルを適用する．

6.　実　　装

6.1　DOM による入力

本システムでは，MXML・HTML を扱う方法として DOM (Document Object Model)[4] を用いる．DOM は XML 形式の文書の要素や属性をノードとし，文書全体を木（DOM 木）の形で扱う．DOM 木を用いることで，XML 内を任意の順番でアクセスでき，柔軟な解析ができる．従って本システムでは，読み込んだ MXML 文書を DOM 木（MXML 木）の形に変換し，これに対して変換を行う．

6.2　Visitor パターンを用いた変換

本システムでは，MXML 木の解析のためにデザインパターンの Visitor パターンを基に実装した `DocumentVisitor` クラスを用いる．このクラスを用いることで DOM のノードを種類ごとに解析することができ，柔軟な変換が可能となる．

特に，ノードが要素を表す場合，`DocumentVisitor` は要素を変換するための `NodeTransformer` の実装クラスのインスタンスを生成する．`NodeTransformer` は各 MXML 要素を，対応する HTML5 の要素に変換するためのインターフェイスであり，1 つの MXML 要素に対して 1 つの `NodeTransformer` の実装クラスが存在する．例として，MXML の `Button` 要素を変換するために，`NodeTransformer` インターフェイスを実装した `Button` クラスが存在する．このように，各要素の変換処理をクラス単位で記述することで，新たな要素変換を追加する際の拡張性の向上に寄与することができる．

また，この `NodeTransformer` の実装クラスのインスタンス生成には，AbstractFactory パターンを基にした `TransformerFactory` クラスを用意した．これは，要素の名前

図 8 Visual Basic 6 の GUI コンポーネント
Fig. 8 GUI components of Visual Basic 6.

表 1 Visual Basic 6 の GUI コンポーネントの対応の状況
Table 1 GUI components of Visual Basic 6.

対応済み	未対応	対象外
ラベル	水平スクロールバー	タイマー
フレーム	垂直スクロール	ディレクトリリスト
チェックボックス	シェイプ	ドライブリストボックス
ピクチャーボックス	ライン	ファイルリスト
テキストボックス		OLE コンテナ
コマンドボタン		データ
オプションボタン		
リストボックス		
コンボボックス		
イメージ		

を入力としてその要素を変換する `NodeTransformer` の実装クラスのインスタンスを生成する．これにより，`NodeTransformer` の実装クラスのインスタンス化を抽象化することができ，柔軟なインスタンスの生成を可能とする．

6.3 DOM による出力

本システムでは，変換された HTML5 要素を用いて DOM による HTML5 の木を構築し，これが最終的な HTML5 文書として出力される．

7. 評　　価

この節では，本システムの変換率の評価を行う．3 節で説明したとおり，ここでは，Visual Basic 6 の機能と比較した評価を行う．

Visual Basic 6 には，ラベルやコマンドボタンなどの GUI コンポーネントが 20 種類存在する（**図 8**）．本研究では，これらの GUI コンポーネントに対応する Flex の標準コンポーネント（Flex UI コンポーネント）を HTML5 要素に変換している．

本システムでは，Flex UI コンポーネントのうち 26 種類が HTML5 に変換可能であるが，Visual Basic 6 の 20 種類の GUI コンポーネントのうちでは 10 種類に対応している（**表 1** 左）．また，表 1 のコンポーネントの中で，対応する Flex UI コンポーネントが存在しないため，本システムでは対象外となるものを表 1 の中央に，対応するものは存在するが現時点で変換できていないものを表 1 の右に示した．ここで，本システムが対応するコンポーネントの内，対応済みのもので変換率を計算すると，62.5%と評価することができる．

表 2 に本システムにより変換可能な Flex UI コンポーネントと HTML5 要素の対応表の一部を示す．変換後の HTML5 で属性が必要なものは表 2 の右段に，属性名と値の関係で示した．

また，現時点で変換不可なコンポーネントがある理由として

表 2 Flex コンポーネントから HTML5 要素への変換
Table 2 Transformation from Flex components to HTML5 elements.

Flex コンポーネント名	HTML5 要素名	HTML5 属性
Button	button	
CheckBox	input	type=“checkbox”
ComboBox	select	
HBox	div	class=“HBox Horizontal”
Image	image	
Label	span	
List	select	multiple=“multiple”
Text	div	
TextInput	input	
VBox	div	class=“VBox Vertical”

は以下のものがある．

(1) HTML5 の要素に対応するものがない．

水平スクロールバー，垂直スクロールバーは HTML5 の場合，要素として，それ単体では対応するものがない．しかし，これらは Web ブラウザの機能として標準で実装されており，自動的にページに付与されるため変換の必要性は低いと考える．

(2) スクリプト要素が絡み解析が複雑なもの．

シェイプやラインなどのコントロールを変換する際には ActionScript によって作成されたスクリプトを解析し，JavaScript などのブラウザ上で動作可能なスクリプトに変換する必要がある．本変換システムではこのようなスクリプトに関わる機能には現時点では対応していない．

8. おわりに

我々は本システムを用いて，Flex アプリケーションを HTML5 の形式に変換することで，iOS 環境で動作可能にする方法を示すとともに，変換例などを通し具体的な結果を示すことができた．しかし，スクリプトの変換が未対応であることなどが今後の課題として挙げられる．今後は，スクリプト間の変換も行うことで，より実用的なアプリケーションの変換を目指したい．

参　考　文　献

1) Adobe Labs: Wallaby, `http://labs.adobe.com/technologies/wallaby/`.
2) Adobe Systems: Adobe Flex, `http://www.adobe.com/jp/products/flex/`.
3) Adobe Systems: Mobile application development with Flex, `http://www.adobe.com/products/flex/mobile/`.
4) W3C: Document Object Model (DOM) Level 3 Core Specification version 1.0, `http://www.w3.org/TR/DOM-Level-3-Core/`.
5) W3C: HTML5 A vocabulary and associated APIs for HTML and XHTML, `http://dev.w3.org/html5/spec/Overview.html`.
6) マイクロソフト：Visual Basic デベロッパーセンター，`http://msdn.microsoft.com/ja-jp/vbasic/`.
7) 早川智一，長谷川慎哉，疋田輝雄：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，情報処理学会第 72 回全国大会 (2010).
8) 早川智一，長谷川慎哉，疋田輝雄：中間表現とフレームワークを用いた Web アプリケーション変換，DPSWS2010 (2010).
9) 早川智一，長谷川慎哉，疋田輝雄：Design and Implementation of Intermediate Representation and Framework for Web Applications, Changchun, CSIE2011 (2011).